A General Model of Adaptive Tutorial Dialogues for Intelligent Tutoring Systems

---

A thesis submitted in partial fulfilment of the

requirements for the Degree

of

Doctor of Philosophy

by

Amali Weerasinghe

Supervising Committee

Senior Supervisor:	Professor Dr. Antonija Mitrovic
Associate Supervisor:	Dr. Brent Martin

Examiners

Professor Dr. Peter Brusilovsky	External Examiner
Dr. Peter Andreae	Internal Examiner
Professor Dr. Antonija Mitrovic	UC Examiner

University of Canterbury
2013

Table of Contents

Contents

Acknowledgments

I would like to thank Professor Antonija Mitrovic for all her support and guidance for this research. Thank you Brent Martin, my associate supervisor, for your helpful advice.

I would like to thank University of Canterbury for providing financial support through University of Canterbury Doctoral Scholarship Scheme.

I would like to thank Prof. Benedict du Boulay, Prof. Gordon McCalla and Prof. Stellan Ohlsson for their valuable contributions towards my research.

I would also like to thank the past and present members of the Intelligent Systems Group at the University of Canterbury specially Atefeh, Amir, David, Jay, Moffat, Martin and Pramudi for providing useful feedback on my research.

Encouragement of my parents during this research is greatly appreciated. My heart-felt gratitude to my husband, Chandana for helping me in every possible way to complete this research.

Finally, thanks to the numerous volunteers from the Department of Computer Science at the University of Canterbury for their participation in the evaluation studies.

This thesis is dedicated

to

my loving parents,

my ever-supportive husband Chandana

and

my loving daughters Ravisha and Tharusha

Abstract

Adaptive tutorial dialogues have been successfully employed by ITSs to facilitate deep learning of conceptual domain knowledge. But none of the approaches used for generating dialogues have been used across instructional domains and tasks. The objective of this project was twofold: (i) to propose a general model that provides adaptive dialogue support in both well- and ill-defined instructional tasks (ii) to explore whether adaptive tutorial dialogues are better than non-adaptive dialogues in acquiring domain knowledge.

Our model provides adaptive dialogue support by identifying the concepts that the student has most difficulty with, and then selecting the tutorial dialogues corresponding to those concepts. The dialogues are customised based on the student's knowledge and explanation skills, in terms of the length and the exact content of the dialogue. The model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them.

We incorporated our model into EER-Tutor, a constraint-based tutor that teaches database design. The effectiveness of adaptive dialogues compared to non-adaptive dialogues in learning this ill-defined task was evaluated in an authentic classroom environment. The results revealed that the acquisition of the domain knowledge (represented as constraints) of the experimental group who received adaptive dialogues was significantly higher than their peers in the control group with non-adaptive dialogues. We also incorporated our model into NORMIT, a constraint-based tutor that teaches data normalization. We repeated the experiment using NORMIT in a real-world class room environment with a much smaller group of students (18 in NORMIT study vs 65 in EER-Tutor study) but did not find significant differences. We also investigated whether our model could support dialogues in logical database design and fraction addition using paper-based methods.

Our evaluation studies and investigations on paper indicated that our model can provide adaptive support for both ill-and well-defined tasks associated with a well-defined domain theory. The results also indicated that adaptive dialogues are more effective than non-adaptive dialogues in teaching the ill-defined task of database design.

Chapter 1

Introduction

Since the invention of computers, they have been perceived as an economical solution to achieve the effectiveness of one-on-one human tutoring, which is an extremely effective form of instruction (VanLehn, 2011). Intelligent Tutoring Systems (ITS) that are capable of customising the learning environment for each learner are a significant step towards utilising computers to enhance the efficiency and effectiveness for masses of learners.

The dialogue between a learner and a teacher facilitates high interactivity, a key characteristic in a one-on-one human tutoring environment. The dialogues also provide opportunities to reflect on the existing knowledge as well as to integrate new knowledge (Chi, Siler, Jeong, Yamauchi, & Hausmann, 2001). There have been several attempts to utilise tutorial dialogues to teach corresponding conceptual knowledge during problem-solving (Aleven & Koedinger, 2002; Aleven, Popescu, Ogan, & Koedinger, 2003; Mitrovic, 2005; Evans & Michael, 2006; Weerasinghe & Mitrovic, 2006; Heffernan, Koedinger, & Razzaq, 2008; VanLehn, van de Sande, Shelby, & Gershman, 2010). However none of the approaches used for generating dialogues have been used to facilitate adaptive tutorial dialogues in multiple domains. In this research we proposed a general model for adaptive dialogues across domains. Our model provides adaptive dialogue support by identifying the concepts that the student has most difficulty with and then selecting the tutorial dialogues corresponding to those concepts. The dialogues are customised based on the student's knowledge and explanation skills, in terms of the length and the exact content of the dialogue.

This introductory chapter presents a high-level overview of the thesis. Intelligent Tutoring Systems are introduced in the next section. We then discuss domain and student models, two key components of ITSs, followed by a brief description of Constraint-Based Modelling (CBM). We then briefly discuss dialogue-based ITSs and the thesis contributions. Finally, a guide to the rest of the thesis is outlined.

1.1 Intelligent Tutoring Systems

The first attempt to use computer-based support to enhance traditional educational environments was Computer Aided Instruction (CAI) systems (Ayscough, 1977; Last, 1979; O' Shea & Self, 1983) which presented the educational material to students in a static form. Every student received the same material but had some control over the navigation through the curriculum. Later systems were capable of providing feedback based on a student's answers. Expected feedback was organised into blocks of information called *frames*, which define both the topic and the feedback to be presented. For instance, correct answers for a set of questions triggered the next frame to be presented. In the case of incorrect answers an alternative screen is presented. These systems could only handle questions with a limited set of solutions, such as 'yes'/'no', multiple choice or numerical solutions.

Although CAI systems managed to achieve modest gains over traditional classroom settings, they were not as effective as human one-on-one tutoring, which can improve students' learning by up to two standard deviations (Bloom, 1984). A human tutor's ability to scrutinize the student's solution, identify their misconceptions and provide customized explanations yields the effectiveness of human one-on-one tutoring (Bloom, 1984). Human tutors are able to identify a student's strengths and weaknesses and customise explanations based on his/her knowledge level. In contrast, CAI systems presented the same material to all students and were unable to keep track of a student's evolving knowledge. Another drawback of these systems is their inability to target the instructional material to a specific audience. This may result in frustration among students because either (i) they are presented with the material they already know or (ii) given problems are too hard for them to solve. This is due to the system's incorrect assumption of a student's competency. Furthermore these systems do not have the capability of providing remedial supplementary exercises to handle a student's misconceptions due to the lack of information about misconceptions.

This prompted researchers to investigate ways how computer-based systems can closely approximate human tutors. The resultant systems are known as Intelligent Tutoring Systems. ITSs are adaptive educational systems: they adapt to each individual student by making inferences about

a student's knowledge based on his/her behaviour. In contrast to CAI systems, ITSs are capable of handling multiple correct solutions.

ITSs have been highly successful in enhancing learning in a variety of domains such as ANDES Tutor (VanLehn et al., 2010) for Newtonian Physics, EER-Tutor (Mitrovic, Martin, & Suraweera, 2007) for conceptual database design, PACT Algebra Tutor (Corbett, Trask, Scarpinatto, & Hadley, 1998) for high-school level geometry, SQL-Tutor (Mitrovic, 1998) for database querying, etc.

1.2 Domain models and student models

The effectiveness of ITS in enhancing learning comes from their adaptivity which is possible due to modelling of the domain and the student. Modelling the domain involves representing the subject matter in such a way that an ITS can use it for reasoning. There are many possible representations such as constraints, production rules and semantic networks. Representation used for a domain model depends partly on how it will be used. The domain model supports a variety of important pedagogical actions including feedback generation, problem selection and instructional material selection and representation.

On the other hand, student modelling focuses on identifying and gathering relevant information to reason about and represent the knowledge level of a student. High-level details a student model holds includes rate of acquisition, level of competence and motivation. A student model holds low-level details such as what problems they have solved and not solved and which concepts they have grasped or failed to grasp. A student model is used to customize the learning environment of an ITS. i.e. the system's current state together with details from the student model are used to drive the pedagogical actions such as feedback generation or problem selection.

The student model is usually related to the domain model in some way. A student model specifies the knowledge level of each student in terms of the knowledge units represented in the domain model.

1.3 Constraint-based modelling

Constraint-based modelling (CBM) is a method for domain and student modelling introduced by Ohlsson (Ohlsson, 1992) based on his learning theory called "Learning from performance errors" (Ohlsson, 1996). According to Ohlsson, we frequently make mistakes when performing a task even if we possess the declarative knowledge necessary for performing the task correctly. This is because we have not internalised the declarative knowledge in our procedural knowledge and therefore may be overloaded with the number of decisions we must make while performing the task. By practicing the task and catching ourselves (or being caught by someone else) making mistakes we modify our procedure to prevent mistakes. With repeated opportunities to perform the task, we internalise all the declarative knowledge resulting in fewer mistakes.

Some student modelling approaches such as model tracing (Anderson, Corbett, Koedinger, & Pelletier, 1996) determine the correctness of a student's actions by comparing them to the correct set of actions specified by the system. In CBM, we are not interested in what the student has done, rather the current state they are in. This is supported by the fact that a student cannot arrive at a correct solution by traversing a state that violates any of the domain principles.

A number of tutoring systems have been developed using the constraint-based methodology by the Intelligent Computer Tutoring Group (ICTG) at the University of Canterbury. They are designed to support individual learning for a number of domains. For example, EER-Tutor facilitates learning conceptual database design, a design task, whereas NORMIT (Mitrovic et al., 2007) teaches data normalization, a procedural task. COLLECT-UML (Baghaei, Mitrovic, & Irwin, 2007) supports collaborative learning, while teaching object-oriented design using the Unified Modelling Language (UML).

1.4 Dialogue-based Systems

The dialogue between a learner and a teacher facilitates high interactivity, a key characteristic in a one-on-one human tutoring environment. Dialogues provide opportunities to reflect on the existing knowledge as well as to integrate new knowledge. There have been several research attempts utilising tutorial dialogues to teach corresponding conceptual knowledge during problem solving

(Aleven & Koedinger, 2002; Aleven, Popescu, et al., 2003; Mitrovic, 2005; Evans & Michael, 2006; Heffernan et al., 2008; Weerasinghe & Mitrovic, 2006; VanLehn et al., 2010). More details of these systems are presented in Chapter 2. However none of the approaches used for generating dialogues have been used in multiple domains to assist students to acquire deep understanding of domain knowledge. If such a general model could be developed, it could potentially be used to explore the differences between teaching and learning different types of instructional tasks. Furthermore, none of the published literature reports any investigations exploring whether adaptive tutorial dialogues (customised based on a student's knowledge level and the interactions with the ITS) are more effective than non-adaptive in enhancing learning in problem-solving environments.

1.5 Overview of the thesis

The goals of this research are two-fold:

(i) To propose a general model that provides adaptive dialogue support for multiple domains. Adaptive dialogue support is the process of customising the dialogues including their selection, content and timing based on a student's knowledge and the history of tutoring sessions. This model emulates the behaviour of human tutors by providing adaptive dialogues during problem solving in response to errors.

(ii) To experimentally evaluate the hypothesis "Adaptive tutorial dialogues are more effective than non-adaptive dialogues in learning both domain knowledge and procedural knowledge" based on our model.

In order to achieve objective (i), we identified the differences between instructional domains as well as between a task and a domain. We propose that two orthogonal dimensions need to be considered: domain, and the task. We believe the two dimensions are continuous; there is a spectrum arranging domains from ill- to well-defined ones, as well as another spectrum for instructional tasks. More details are given in Section 3.1.

Based on this categorization, we proposed a model that supports adaptive tutorial dialogues in both ill- and well-defined tasks. Our model provides adaptive dialogue support by identifying the concepts that the student has most difficulty with and then selecting the tutorial dialogues corresponding to those concepts. The dialogues are customised based on the student's knowledge

and explanation skills, in terms of the length and the exact content of the dialogue. The model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all error types in a domain. At the lowest level, an error type is associated with one or more violated constraints, which form the knowledge base of a constraint-based tutor. The error types are then grouped into higher level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When a student solution has multiple errors, the hierarchy is traversed to select the pedagogically most suitable error for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue, learners are able to provide answers by selecting the correct explanation from a list.

The three highest levels of the error hierarchy (the first component of the model) are domain-independent. Further divisions of these nodes are associated with domain-specific concepts. Even though dialogues consist of domain-specific prompts, the structure is domain-independent. Adaptation rules (the last component) which customise dialogues are domain-independent except for the time period of inactivity the tutoring system waits before intervening.

As the first step towards achieving goal (i), we incorporated our model into EER-Tutor. As the second step, NORMIT was enhanced to facilitate tutorial dialogues using our model. We also investigated whether our model could be used in the domains of logical database design and fraction addition using paper-based methods. Therefore we have tested the applicability of our model in four different domains: (i) conceptual database design (ii) data normalization (iii) logical database design and (iv) fraction addition. All these tasks are associated with a well-defined domain theory. All the tasks except conceptual database design are well-defined.

To test hypothesis related to goal (ii), two evaluation studies were conducted in authentic classroom environments: the first one involved dialogue-based EER-Tutor, whereas the second, the dialogue-based NORMIT. The results revealed that the acquisition of domain knowledge (represented as constraints) of the experimental group who received adaptive dialogues was significantly higher than their peers in the control group with non-adaptive dialogues even when both groups spent a similar amount of time with EER-Tutor. The experimental group's problem-solving performance also increased significantly more than the control group. The second study

involved a much smaller group of students (18 in NORMIT study vs 65 in EER-Tutor study) and did not find significant differences. (i.e. even though the learning curves indicated that the rate of learning by the experimental group is higher than that of the control group, it was not significant. Both groups also learnt a similar number of constraints).

1.6 Guide to the thesis

Chapter 2 provides the background for this research by introducing Intelligent Tutoring Systems and constraint-based modelling. It also includes a survey of currently available dialogue-based systems. We discuss our solution, a general model that provides dialogue support across domains, in Chapter 3. The studies conducted to evaluate the effectiveness of our model and their results are presented in Chapter 4. Finally, Chapter 5 outlines the conclusions and discusses future directions for research in this area.

During this research project, we have prepared and presented ten publications, which are listed in Appendix E.

Chapter 2

Background

Human one-on-one tutoring is widely believed to be the most effective form of tutoring leading to large learning gains confirmed by experimental evidence (Bloom, 1984). Ever since computers were invented, researchers have been exploring ways of utilizing them as a means of enhancing the efficiency of learning. The first attempt was Computer Aided Instruction systems (Ayscough, 1977, Last, 1979, O' Shea & Self, 1983) which presented the educational material to students in a static form. Every student received the same material but had some control over the navigation through the curriculum. Another attempt was "branching", where the systems' response was based on student's answers. Expected feedback was organised into blocks of information called frames, which define both the topic and the feedback to be presented. For instance, correct answers for a set of questions triggered the next frame to be presented. In the case of incorrect answers, an alternative screen is presented. These systems could only handle questions with a limited set of solutions, such as 'yes'/'no', multiple choice or numerical solutions. Although CAI systems managed to achieve modest gains over traditional classroom based teaching (Kulik, Kulik, & Cohen, 1980), they still failed to realise the dream of utilizing computers to achieve the effectiveness of human one-on-one tutoring. Intelligent Tutoring Systems are the result of research attempts to develop computer-based systems that can closely approximate human tutors. These systems have the ability to reason dynamically about a student's knowledge and provide a customised learning experience to each student based on his/her behaviour. The domain knowledge that the ITSs are designed to teach is represented as a dynamic model with a set of rules which governs the way the system reasons. In contrast to CAI systems, ITS can handle a set of correct solutions.

ITSs have been developed for a variety of domains for different types of instructional settings. For example the focus of some tutors are individual learning while some are for collaborative learning. Model-tracing tutors (Anderson et al., 1996) provide problem-solving environments which guide students towards the correct solution through immediate feedback. Simulation-based

tutors teach the target domain using simulation environments (Alexe & Gescei, 1996; Munro et al., 1997). The focus of collaborative tutors (Dillenbourg & Self, 1992) is to facilitate positive interaction among learners by encouraging participation and supporting collaborative problem-solving. In this research our focus is on tutoring systems that support individual learning by solving problems. Similar to model-tracing tutors, they provide problem-solving environments in which student receive customised feedback as they progress towards correct solutions. These tutors allow students to work at their own pace and request feedback when they deem necessary.

We discuss the different components of a typical ITS in Section 2.1. A description of the constraint-based methodology and some examples of existing constraint-based tutors is presented in Section 2.2. Finally we present details of some existing significant dialogue-based systems in Sections 2.3 and 2.4.

2.1 Intelligent Tutoring Systems

The architecture of a typical ITS (Figure 2.1) consists of four main components: a domain module, a student modeller, a pedagogical module and an interface.

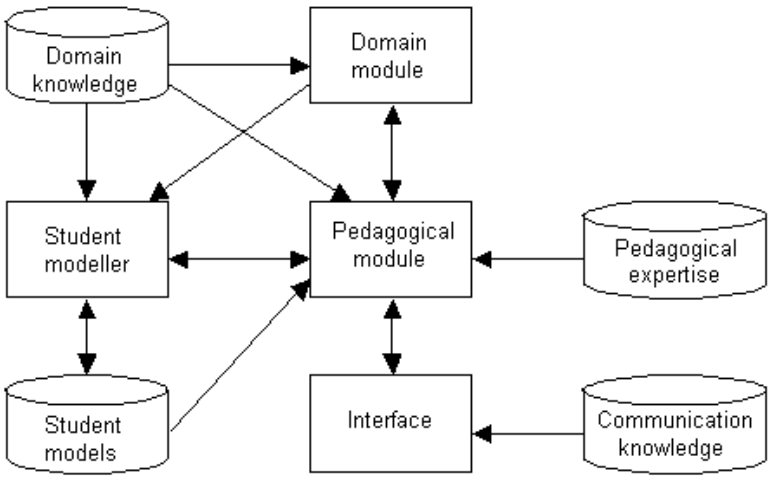


Figure 2.1: ITS Architecture

The student interacts with the ITS through the interface. Depending on the implementation, the system waits for a student's request to evaluate his/her solution or provides immediate feedback after tracing a student's behaviour. Evaluating a solution may result in a number of actions within the system. The student modeller compares the student's solution against the system's solution using the domain module. The student modeller then updates the student model to reflect the student's new knowledge. The pedagogical module then provides feedback or selects a new problem based on the student's performance. We now discuss each of the four main components.

2.1.1 Domain Module

The domain module contains all the required declarative or procedural knowledge or both for a domain. The extent of domain knowledge could vary from the expert knowledge required for solving problems to a subset of such knowledge required for teaching purposes. Some domain models are capable of generating the correct solution based on an individual student's problem-solving path. The domain model of PACT Algebra Tutor (Corbett et al., 1998) is one such model that is capable of following a student's actions and generates the correct solution. In contrast, some domain models focus only on pedagogically informative states, rather than the procedure that a student takes to arrive at the answer. The domain model of SQL-Tutor (Mitrovic et al., 2007) evaluates a student's attempt using these states and compares the student's attempt to the system's stored ideal solution.

The domain module may take many forms, depending on the domain it represents, the knowledge representation used and the granularity of the information represented. In cognitive tutors, the domain level consists of low-level production rules that completely describe the expected student behaviour down to atomic thought components (Anderson & Lebiere, 1998). Constraint-based systems (Mitrovic et al., 2007) capture the domain principles that need to be satisfied by correct solutions. In page-based systems such as adaptive hypertext (Brusilovsky, 2000), domain knowledge is stored at the page level, and provides basic information about the content of the page, which is used in problem selection and instructional material sequencing.

2.1.2 The Student Modeller

The student modeller evaluates a student solution and maintains a model of a student' knowledge and skill levels. This representation is known as the *student model*. It includes long-term knowledge such as the student's domain mastery, as well as short-term knowledge, such as whether or not the student violated a rule in his/her most recent attempt. There is increasing interest in modelling meta-cognitive skills such as the self-explanation, reflection and help-seeking strategies as well as affective states such as motivation in the student model (Aleven & Koedinger, 2000).

The student modeller is one of the most important modules in an ITS. If the student models do not provide a close approximation of the student's knowledge, the quality of the pedagogical decisions would be affected. Therefore, researchers are interested in discovering student modelling techniques that are capable of modelling both long-term and short-term knowledge of a student.

Model tracing (Anderson et al., 1996) (MT) and CBM (Ohlsson, 1994) are two popular short-term student modelling techniques. MT deals with procedural knowledge of a domain whereas CBM focuses on declarative knowledge. Student models can be generated using either a top-down approach or a bottom-up approach. Cognitive tutors (Anderson et al., 1996), which trace the student's solution path using knowledge of all legal paths, generate student models using the bottom-up approach. Constraint-based tutors also use a bottom-up approach, but do not trace the student's solution in a step-by-step manner. They only evaluate the submitted solution.

Overlays (Holt, Dubs, Jones, & Greer, 1994) and stereotypes (Rich, 1989) are two widely used techniques to model long-term student knowledge. Overlays represent a student's knowledge as a subset of the domain knowledge. It starts with an empty model assuming that a student does not possess any knowledge of the domain initially. The model is populated as the student interacts with the system. Hence overlays use a bottom-up approach. In contrast, stereotypes start modelling the student by classifying the student into levels of expertise, usually based on the performance of a pre-test. Thus overlay models use a top-down approach to generate student models. Many systems use stereotypes or overlay models in conjunction with either MT or CBM to represent students' knowledge. For example, CBM stores the history of each constraint (i.e. whether the student has been able to successfully apply the constraint in each occasion) in its overlay model for each

student. Cognitive tutors use the technique known as knowledge tracing (Koedinger & Aleven, 2007) to compute the probability that the student knows each knowledge component represented by a production rule, in its overlay model.

2.1.3 Pedagogical Module

The pedagogical module, the component that makes decisions about the teaching of the domain, is the driving engine of an ITS. Pedagogical decisions vary from low-level decisions (such as deciding what type of feedback to present to the student in a particular situation in response to a student's action) to high-level decisions (such as selecting the next topic for the student). The pedagogical module uses information from the student, domain and tutoring models to arrive at each decision.

Pedagogical strategies can vary from traditional didactic approaches to exploratory learning. The didactic approach focuses on instructing the learner during learning (Anderson, 1993). This approach is more suitable for novices who need considerable guidance than more knowledgeable students who find it restrictive. Tutoring systems that employ this approach initiate and control student activity. Such systems work well for well-defined tasks for which there is a clearly defined procedure from the problem statement to the correct solution. PACT Algebra Tutor is an example of a system which monitors a student's solution path in a step-by-step manner and provides immediate feedback whenever the student strays from the ideal path. In contrast, exploratory learning encourages learning from experience (Lesgold, 1987; Shute, Glaser, & Raghavan, 1989) and promotes new knowledge to be constructed through reflection, self-explanation and induction. This approach focuses on giving total freedom to explore the domain. Even though the more knowledgeable students are generally able to engage in deep learning using this approach, novices might need considerable time to achieve the pedagogical objectives. Tutoring systems that employ this strategy work well for ill-defined tasks for which there is no clearly defined procedure to arrive at the correct solution. For example, KERMIT(Suraweera & Mitrovic, 2004; Mitrovic et al., 2007), an ITS that teaches the ill-defined task of conceptual database design, allows students to develop the solution in any order they like and to decide when they need feedback.

2.1.4 Interface

The interface is the medium between the student and the tutoring system. Typically the student uses the interface to solve problems and the system presents the feedback through the interface. There have been many research attempts to identify characteristics of efficient interfaces that will enhance the user perception of a system in order to improve learning. Firstly, an interface designed to reduce a student's working memory load is an important aspect of an ITS. One method to achieve this is to make available all the components of the problem that are not part of teaching through the interface. The interface of KERMIT minimises the working memory load of the student by providing both the current problem description and the toolbar of ER constructs in one view (Figure 2.2)[1]. This allows the student to concentrate on the actual modelling problem at hand. When creating ER diagrams in KERMIT, students are also forced to highlight the text in the problem statement that corresponds to each new construct they model in their solution. This enables the student to keep track of what they have modelled, reducing the complexity of the problem-solving process. Secondly, an interface should facilitate visualising the goal structure for solving the problem helping students towards task completion. Finally, the interface should be able to motivate students as student motivation is vital for continued use of the system leading to mastery of the domain.

2.2 Constraint-based modelling

CBM (Ohlsson, 1992) is a method for domain and student modelling introduced by Ohlsson based on his learning theory. This section outlines the theory followed by a discussion of how CBM is used to develop intelligent tutoring systems.

[1] Figure 2.2 represents the interface of KERMIT-SE, which is very similar to that of the original of KERMIT. The only difference is that bottom window of KERMIT has only the section that lists All Errors.

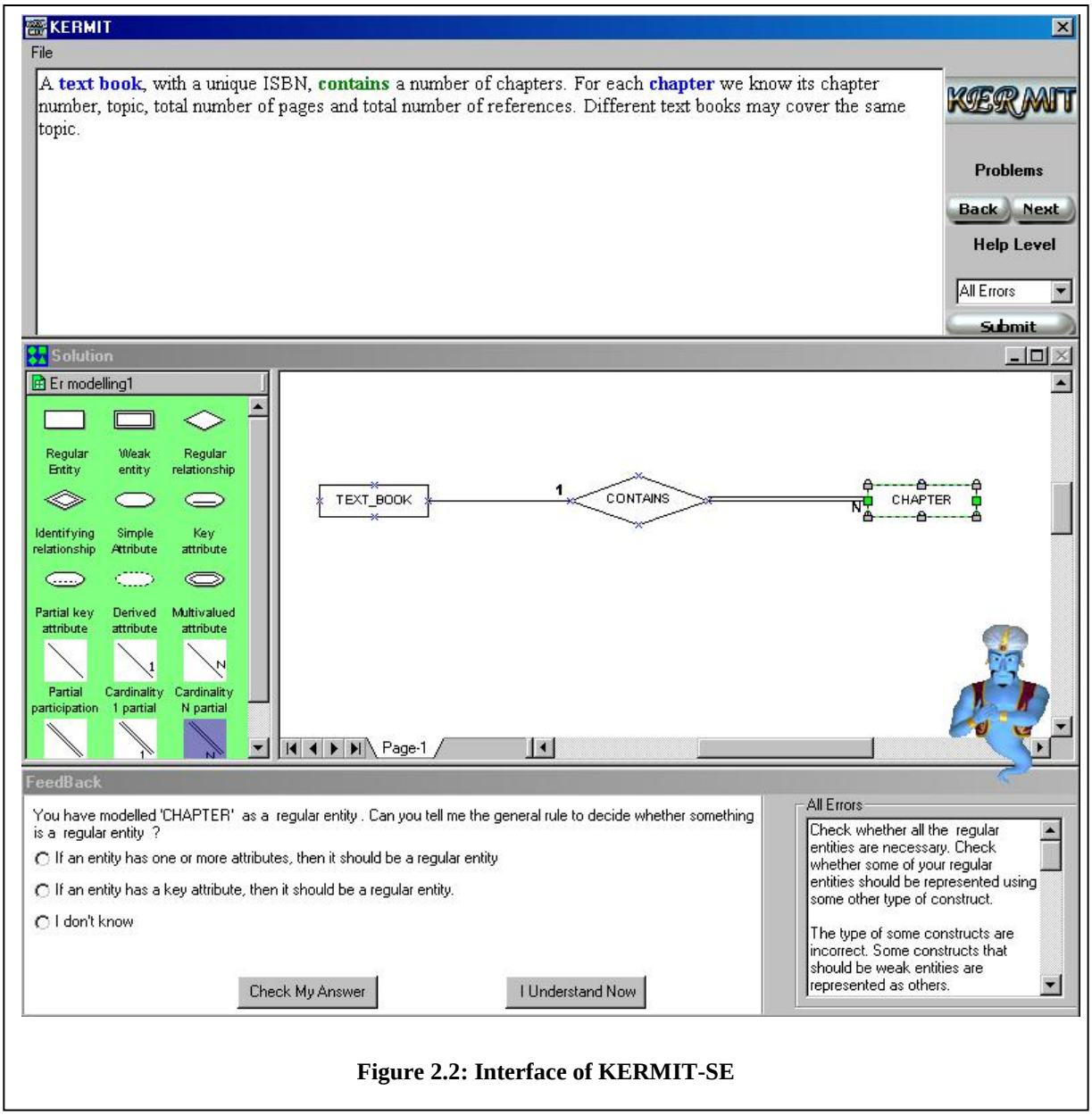


Figure 2.2: Interface of KERMIT-SE

2.2.1 Learning from Performance Errors

CBM is based on Ohlsson's learning theory called "Learning from performance errors" (Ohlsson, 1996). The theory states that we learn when we catch ourselves (or are caught by some other party) making mistakes. Furthermore, we make mistakes even if we possess the required declarative knowledge because we may be overloaded having to consider too many possibilities in any given

situation. Thus learning declarative knowledge is not sufficient to make the correct choice; we also need to learn how to apply the declarative knowledge.

Ohlsson uses constraints to represent how declarative knowledge can be applied to a given situation. Each constraint is an ordered pair $<C_r, C_s>$, where C_r is the relevance condition and C_s is the satisfaction condition. The first condition specifies when a piece of declarative knowledge is relevant, while the second one describes the state whereby the piece of knowledge has been correctly applied.

In other words,

IF <relevance condition> is true

THEN <satisfaction condition> should also be true

Consider a person from United States (right-hand side driving) starting to drive a car in a New Zealand (left-hand side driving). One of the factors he/she has to consider is whether the road system is designed for right-hand side driving or left-hand side driving. At the beginning of his/her driving, he/she has to make a decision whether to steer the car on to the right-hand side of the road or the left-hand side. The following constraint reflects this situation:

If driving in New Zealand

You better be on the left-hand side of the road

This constraint is relevant for all situations of driving in New Zealand. According to Ohlsson's theory, a driver who is knowledgeable about the left-hand driving rule may still steer the car to the right because he/she still new to the situation. He/she internalises the constraint only after catching themselves violating it or being reminded by someone else. However, a driver who is familiar with left-hand side driving, may "intuitively" steer the car to the left hand side of the road due to repeated applications of the constraint. An expert driver has this constraint internalised as procedural knowledge.

2.2.2 Domain Models in Constraint-based Tutors

The domain model of a constraint-based tutor consists of a set of constraints on correct solutions. They represent only the declarative knowledge of a domain. Thus constraints can be used to identify correct solutions from the space of all possible solutions. CBM is based on the observation that no correct solution violates any fundamental ideas or concepts of the domain. Incorrect solutions can be identified as solutions that do not adhere to the semantic and syntax rules of the domain.

If a constraint is relevant to the student's solution and its satisfaction condition is violated, the domain concept represented by the constraint needs to be taught to the student. Violation of a constraint by a student indicates a misunderstanding of a domain concept that needs to be corrected. Once constraint violations are identified, pedagogical actions could be used to correct a student's misconceptions.

For example, a constraint that applies to the right-most column in multi-column addition be written as shown in Figure 2.3.

Relevance condition (C_r):	The student has specified the right-most column in the addition problem and the sum of the two integers in the right-most column does not exceed 10
Satisfaction condition (C_s):	The answer for the right-most column = the sum of the two integers in the right-most column

Figure 2.3: A constraint for performing multi-column addition

Initially, the student solution is matched against the relevance condition. The satisfaction condition is evaluated only if the relevance condition is met; otherwise the constraint is ignored. The constraint is considered satisfied if the student solution satisfies the C_s, else the constraint is violated.

CBM is computationally simple because student modelling is reduced to pattern matching (Ohlsson, 1994). During the evaluation of a problem state, all relevance conditions are matched against the problem state. In a case where the solution matches the relevance condition, it is then checked against the satisfaction condition. If the satisfaction condition is not met, then the

constraint is violated, which indicates an error. Furthermore, an existing algorithm can merge constraints into a unified structure called a RETE network, which can increase the efficiency of constraint matching (Mitrovic, 1998).

Because student modelling using CBM is reduced to pattern matching, an ITS that uses CBM does not require a runnable expert module to generate pedagogical actions. This is an important advantage, as it is difficult to design an expert module for many domains. Moreover, CBM also does not require extensive bug libraries, which enumerate students' misconceptions about the domain.

One other advantage of CBM is its ability to identify multiple correct solutions. The model-tracing approach requires enumerating all possible correct solution paths in order to identify multiple correct solutions. However, constraints can be used to identify all possible correct solutions without enumerating them. This is a very important advantage because a number of studies have suggested that students rapidly switch between several different strategies during problem-solving (Ohlsson, 1994). Ohlssson (1994) defines this as the *radical strategy variability* phenomenon, and if it turns out to be the normal case, then it invalidates approaches that assume the student follows only a single solution path.

Knowledge acquisition for constraint-based tutors is considerably less demanding compared to cognitive tutors. Mitrovic reported that she required on average 1.1 hours (Mitrovic & Ohlsson, 1999) to develop a constraint, which is significantly less time compared to ten or more hours required by cognitive tutors per production rule (Anderson et al., 1996).

2.2.3 Constraint-based tutors

A number of tutoring systems based on CBM have been developed by the Intelligent Computer Tutoring Group (ICTG) at the University of Canterbury. They cover a variety of domains including domains focusing on computer programming tasks, procedural tasks and design tasks. All these systems have been implemented as practice environments, where students are given numerous opportunities to learn through problem-solving. The student can ask for assistance from the system anytime during the problem-solving process.

Constraint-based tutors provide assistance to students of a variety of age groups. While the majority of constraint-based tutors are designed for university-level students, some tutors cater for younger students. CAPIT (Mayo, Mitrovic, & McKenzie, 2002), the punctuation tutor, was developed to assist 9-11 year old school children to learn capitalization and punctuation skills. LBITS (Martin & Mitrovic, 2002) is another constraint-based tutor developed to teach basic English language skills to elementary and secondary school students. It uses a series of "puzzles" such as synonyms, crosswords and plurals to teach these skills.

A suite of tutoring systems have been developed to teach database concepts such as conceptual database design, data normalization and database querying (Mitrovic et al., 2007). The database suite includes EER-Tutor (Mitrovic et al., 2007) for learning conceptual design, ERM-Tutor (Milik, Marshall, & Mitrovic, 2006) for logical database design, NORMIT (Mitrovic et al., 2007) for data normalization and SQL-Tutor (Mitrovic et al., 2007) for SQL database query language. We now briefly discuss the domain of database design for which several constraint-based tutors have been developed. Outlines of some of these systems are presented next.

The domain of database design: Learning how to develop good quality databases is a core topic in today's Computer Science curriculum throughout the world. Bartini et al. (1986) define database design to be the task of "designing the structure of a database in a given environment of users and applications such that all users' data requirements and all applications' process requirements are 'best satisfied'". The process involves four stages:

(i) Requirements specification
(ii) Conceptual design
(iii) Logical design
(iv) Physical design

Requirements specification involves identifying the information needs of various users or groups for whom the database is to be developed. The conceptual design phase models the users' and applications' views of information and may include a specification of the processing or use of the information (Bartini, Lenzerini, & Navathe, 1986). The goal of this stage is to produce a formal and accurate representation of the database requirements that is independent of any database management system (DBMS). The conceptual model is then translated into the logical model of

the chosen DBMS, such as a relational data model, during the logical design phase. Finally, the refined logical data model is transformed into a form suitable for the specific DBMS during the physical design phase.

The quality of conceptual schemas is vital for database systems. ER modelling, originally proposed by Chen (1976), is one of the most widely used database modelling techniques to teach conceptual database design. The ER model views the world as consisting of *entities* and *relationships* between them. Entities may be physical or abstract objects, events, roles played by people or anything else data should be stored about. Each entity is described in terms of its important features, called *attributes*. Relationships represent various associations between entities, and also may have attributes.

We will now discuss the process of designing a database using a simple example. Now consider the following problem statement (Figure 2.4) a student is given.

Design a database for a hotel chain. Each hotel has a name, code, address and a category (the number of stars). The total number of beds is known. Each hotel has many rooms, and each room has a number. Rooms may be of two types (single or double). Some rooms have TVs. For each employee, his/her employee number should be stored, as well as the name, address, home phone number, job position and salary. Each employee works for just one hotel. Each hotel has a manager and a manager can manage at most one hotel.

Figure 2.4: A problem statement in conceptual database design

From this statement, it is clear that hotels, rooms, employees and managers are of importance. The student may start by drawing the entities first. However he/she needs to understand managers are also employees and a separate entity is not needed. The student also needs to understand that room numbers are unique only within the hotel, and it should be modelled as a weak entity type. Problems in this domain are often ambiguous and incomplete and students reason about the requirements and use his/her world knowledge to make valid assumptions. For example, student must be able to identify that EMPLOYEE and HOTEL should be modelled as regular entity types as each has its own unique identifier known as a *key attribute* (i.e. Code is the key attribute for HOTEL and employee number for EMPLOYEE).

Each entity type is described in terms of some attributes. For example, each hotel would be described by name, code, address and a category. All these attributes are explicitly stated in the problem statement. For each room, we need to know the number and type (number of beds). Since the statement indicates that only some rooms have TVs, we need an attribute to specify whether a TV is included in the room. Finally, for each employee we need to record name, address, home phone number, job position and salary.

The student also needs to identify the relationships between these three types of entities. Each employee works for just one hotel, which is mentioned explicitly in the statement. However, some relationships are not explicitly stated: the relationship MANAGES between HOTEL and EMPLOYEE indicating that managers are also employees. The student is expected to use his/her world knowledge to identify this relationship.

Once all the concepts are identified, the integrities of the model need to be identified. Each entity type must have at least one key attribute, which uniquely identifies it. As mentioned earlier, the key attribute for HOTEL is code. For EMPLOYEE the key attribute is employee number. As ROOM is a weak entity, it should have a partial key, which is used to identify each room uniquely when combined with the key attribute of HOTEL. The partial key for ROOM is room number.

An ER schema is usually presented in graphical form, and Figure 2.5 represents the ER diagram for the hotel database. The student is also expected to define two types of integrities defined on relationships: *participation* and *cardinality*. Participation indicates whether an entity type participates in a relationship totally (indicated by the double line in the diagram) or partially (shown by the single line). The participation between HOTEL and MANAGES is total to indicate that every hotel has a manager. On the other hand, the participation between EMPLOYEE and MANAGES is partial because only some employees are managers. This is another piece of information that is not explicitly stated in the problem statement.

The second type of integrity, cardinality, specifies the number of instances of the relationship that an entity instance can participate in (shown by *1*and *N* in Figure 2.5). The cardinality between HOTEL and WORKS_FOR is 1 indicating that an employee can work for at most one hotel. As a single hotel has many employees the cardinality between EMPLOYEE and WORKS_FOR is N. The complexity could be further increased by relationships possibly involving more than two entity types (higher degree relationships), and having simple, composite or multivalued attributes.

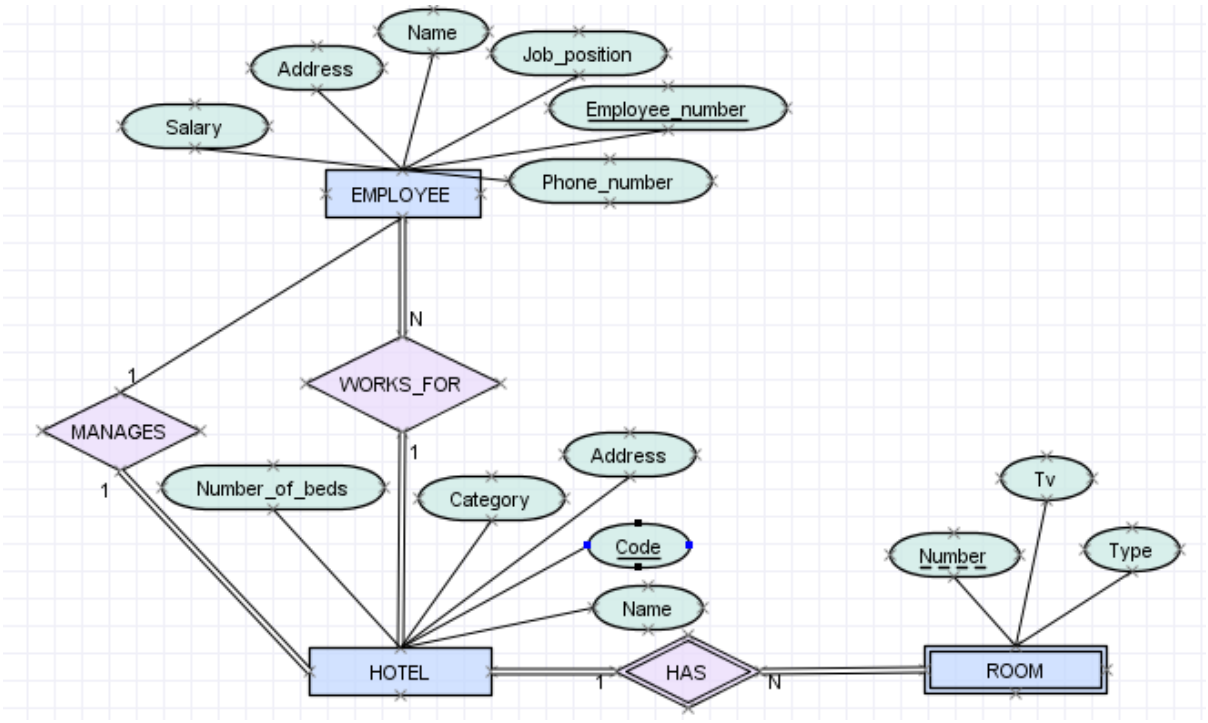


Figure 2.5: Data model for the HOTEL database

As evident from this simple case, there are many things that the student has to know and think about when developing an ER diagram. The student must understand the different aspects of the data model used: the basic building blocks available as well as the integrity constraints specified on them. In real-world situations, the problem statement would be much longer, often ambiguous and incomplete. Making valid assumptions is vital to identify the integrities. The student is expected to make valid assumptions by reasoning about the requirements and using his/her own world knowledge. There is no algorithm to use to derive the ER schema for a given set of requirements. There is no single, best solution for a problem, and often there are several correct solutions for the same requirements.

KERMIT: Knowledge-based ER Modelling Intelligent Tutor: KERMIT is an ITS designed to assist university-level students learn conceptual database design. For a detailed discussion of the system, see (Suraweera & Mitrovic, 2004); here we present some of its basic features. KERMIT provides a problem-solving environment in which students can practice database design using the Entity Relationship (ER) data model. The system is intended to complement traditional instruction, and assumes that students are familiar with the ER model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions, and a student modeller, which analyses student answers and generates student models.

KERMIT contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the correctness of the student's solution, KERMIT compares it to the correct solution, using domain knowledge represented in the form of 92 constraints (Suraweera & Mitrovic, 2004) .

Consider one of the problems in KERMIT (Figure 2.6). Figure 2.7 presents a student solution and a correct solution for the problem statement in Figure 2.6. The student solution contains multiple errors: (i) CHAPTER should be modelled as a weak entity; (ii) CONTAINS should be modelled as an identifying relationship; (iii) TEXT_BOOK should have at least one key attribute; (iv) TEXT_BOOK should have at least one attribute; (v) CHAPTER should have a partial key; (vi) CHAPTER should have at least one attribute and (vii) participation between TEXTBOOK and CONTAINS should be total.

Each text book has a unique ISBN (International Standard Book Number), and contains several chapters. Each chapter has a chapter number (unique within a book), the number of pages and the number of references. A chapter covers a single topic, but the same topic may be covered in various books.

Figure 2.6: A problem in KERMIT

When the student solution in Figure 2.7(a) is submitted to KERMIT, one of the constraints that will be violated is constraint 11. It checks for the existence of a matching regular entity in the ideal solution for each regular entity in the student solution. This is violated due to the non-existence of a matching regular entity for CHAPTER in the ideal solution. In other words,

CHAPTER should be modelled as a weak entity. Violation of this constraint will initiate the feedback message *Check whether all the entities are necessary*. *Check whether some of your regular entities should be represented using some other type of construct*, which appears on the bottom right window of the interface (Figure 2.2).

Figure 2.8 presents constraint 11 in pseudo-code form whereas Figure 2.9 presents its implementation in KERMIT. It specifies that for each regular entity in the student solution there should be a matching regular entity in the ideal solution. The relevance condition (`relCond`) checks for the existence of a regular entity in the student solution. The satisfaction condition (`satCond`) checks whether there is a matching regular entity in the ideal solution. This constraint becomes relevant for the student solution in Figure 2.7(a) as it contains two regular entities TEXT_BOOK and CHAPTER. TEXT_BOOK satisfies the satisfaction condition as there is a matching regular entity in the ideal solution. However, CHAPTER violates the satisfaction condition due to the non-existence of a matching regular entity in the ideal solution.

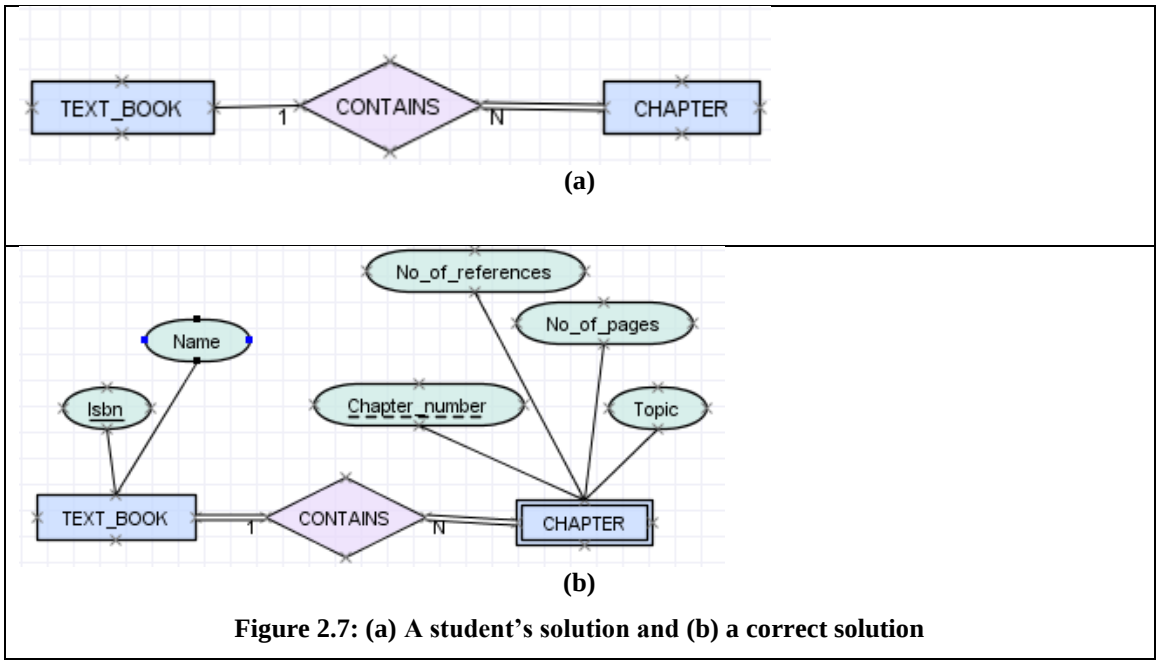


Figure 2.7: (a) A student's solution and (b) a correct solution

In addition to these two conditions, each constraint contains three messages (*feedBack*, *feedBack1*, *feedBack2*) that are used to provide feedback when the constraint is violated. The first message is general, and used as a hint. The other two messages are used as templates for

generating detailed and specific feedback messages. During the generation of feedback, the <viol> tag embedded in the message is customised with the names of the constructs that have violated the constraint. In the example given in Figure 2.7(a), the <viol> takes the value CHAPTER. The message feedBack1 is used when a single construct has violated the constraint, whereas feedBack2 is used when more than one construct have violated the constraint. The types of constructs that violate the constraint are specified in the *Construct* attribute (Figure 2.9). In this example, the type of violated constructs can only be entities (denoted by *ent*). The construct attribute is used for generating a very general feedback message that specifies the type of construct that contain errors. The value '1' assigned to *concept ID* means this constraint is associated with the concept of regular entities. Each constraint is associated with one of the fourteen domain concepts of conceptual database design (Suraweera & Mitrovic, 2004). The conceptID is used to identify the concept that a student has most difficulty with and is used for problem selection.

```
Relevance Condition:    For each regular entity in the student
                        solution
Satisfaction Condition: There should be a matching regular entity in
                       the ideal solution
```

Figure 2.8: Constraint 11 represented in pseudo-code form

Figure 2.2 presents the interface of KERMIT-SE (further discussed in Section 2.4), which is very similar to that of the original of KERMIT. The only difference is that bottom window of KERMIT has only the section that lists All Errors. The interface is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting feedback level. The middle window is the main working area. Students draw ER diagrams in this window.

```
id = 11
relCond = "each obj SSE (= type (obj) e)"
satCond = "each obj RELVNT (and (notNull (matchIS (obj)) (= type (matchIS
(obj)) e))"
feedBack = "Check whether all the entities are necessary. You have extra
regular entities. Check whether some of your entities should be
represented using some other type of construct."
feedBack1 = "<viol> should not be an entity. It may be extra or you may
want to represent it using another type of construct."
feedBack2 = "<viol> should not be entities. They may be extra or you may
want to represent them using other types of constructs."
Construct = "ent"
conceptID = 1
```

Figure 2.9: Implementation of Constraint11 in KERMIT

The feedback from the system is grouped into six levels according to the amount of detail provided: *Correct*, *Error Flag*, *Hint*, *Detailed Hint*, *All Errors* and *Solution*. The first level of feedback, Correct, simply indicates whether the submitted solution is correct or incorrect. The Error Flag indicates the type of construct (e.g. entity, relationship, etc.) that contains the error. For example, when the solution in Figure 2.7(a) is submitted, Error Flag provides the message *Check your entities, that's where you have some problems*. This is associated with the error CHAPTER being modelled as a regular entity instead of a weak entity. The Hint and Detailed Hint offer a feedback message corresponding to the first violated constraint. In the case of student solution in Figure 2.7(a), Hint provides a general message *Check whether all the regular entities are necessary. Check whether some of your regular entities should be represented using some other type of construct*. On the other hand, Detailed Hint provides a more specific message *CHAPTER should not be an entity. It may be extra or you may want to represent it using some other type of construct*, where the details of the erroneous object are given. Not all detailed hint messages give the details of the construct in question, since giving details on missing constructs would give away solutions. A list of feedback messages on all violated constraints is displayed at the All Errors level (as indicated in the bottom right hand in Figure 2.2). The ER schema of the complete solution is displayed at the final level (solution level).

Initially, when the student begins to work on a problem, the feedback level is set to the Correct level. As a result, the first time a solution is submitted, a simple message indicating whether or not the solution is correct is given. The level of feedback is incremented with each submission until the

feedback level reaches the Detailed Hint level. In other words, if the student submits the solution four times, the feedback level would reach the detailed hint level, thus incrementally providing more detailed messages. Automatically incrementing the levels of feedback is terminated at the Detailed Hint level to encourage the student to concentrate on one error at a time rather than all the errors in the solution. The system also gives the student the freedom to manually select any level of feedback according to their needs. In the case of several violated constraints, and the level of feedback is different from "All Errors", the system generates the feedback on the first violated constraint. The constraints are ordered in the knowledge base by a human teacher, and that order determines the order in which feedback is given.

Students have several ways of selecting problems in KERMIT. They may work their way through a series of problems, arranged according to their complexity. The other option is a system-selected problem, when the pedagogical module selects a problem for the student on the basis of his/her student model.

KERMIT maintains two kinds of student models: short-term and long-term ones. The short-term model consists of the satisfaction and violation details of each constraint, identified during the evaluation of the student solution. The short-term model is only dependent on the submitted solution and does not account for the history of the constraints such as whether a particular constraint was satisfied during the student's last attempt. The pedagogical module uses the short-term model to generate feedback to the student. The long-term model of KERMIT is implemented as an overlay model. In contrast to the short-term model, the long-term model keeps a record of each constraint's history. It records information on how often the constraint was relevant for the student's solution and how often it was satisfied or violated. The pedagogical module uses the long-term model for problem selection for each student.

The effectiveness of KERMIT in teaching conceptual database design was evaluated in August 2001 during regular lab sessions at the University of Canterbury (Suraweera & Mitrovic, 2004). Sixty-two volunteers who participated in the study were enrolled in an introductory database course. They were randomly assigned to use the complete version of KERMIT (the experimental group) or a cut-down version of KERMIT that only provided the final solution (the control group). Performance in the pre/post-tests revealed that students who used KERMIT attained significantly higher gains ($t = 3.07$, $p < 0.01$) than their peers. The effect size of the

experiment, which allows the comparison of the results of one pedagogical experiment to another, was 0.63. The power of the experiment was 0.75 at significance 0.05, indicating that there is a high probability that the experiment would produce significant results for the same design, the same number of participants and the same effect size.

EER-Tutor: EER-Tutor is a complete web-based re-implementation of KERMIT by a team of software developers whereas KERMIT was the result of a M.Sc. project. The constraint base of KERMIT focuses only on the domain concepts of ER modelling whereas EER-Tutor covers Enhanced ER modelling as well. Enhanced ER modelling enables the user to model specialisations using subclass-superclass and category-subclass relationships (Elmasri & Navathe, 2010) . The back-end of EER-Tutor is implemented in Lisp whereas the diagramming space is a Java applet. KERMIT is implemented using MS-Visual Basic while the diagramming space uses MS-Visio. Furthermore the two constraint bases are different as the constraint base of EER-Tutor was developed independently to that of KERMIT. The constraint base of KERMIT consists of 92 constraints (Suraweera & Mitrovic, 2004) whereas EER-Tutor has 212 constraints (Mitrovic et al., 2007).

Figure 2.10 presents how constraint 11 (Figure 2.9) that checks the existence for a matching regular entity in the ideal solution for each regular entity in the student solution, is implemented in EER-Tutor. As can be seen, the constraint specification language used in EER-Tutor is different from KERMIT. In EER-Tutor, each constraint has a unique identifier, a hint message, a relevance condition, a satisfaction condition, type of construct that the constraint focuses on and the construct to highlight as part of providing detailed feedback. The relevance condition checks whether there is a matching entity (regular or weak) in the ideal solution for each regular entity in the student solution. The satisfaction condition checks whether the type of the entity identified is regular. The phrase “entity types” indicates that this constraint deals with entity types. The text (?tag) identifies the construct to be highlighted.

```
(11
     "Are all regular entity types you have specified necessary? Check
whether some of them should be represented using some other type of
construct."
     (and (match SS ENTITIES (?* "@" ?tag ?l1 "regular" ?*))
          (match IS ENTITIES (?* "@" ?tag ?l2 ?type ?*)))
     (test IS ("regular" ?type))
     "entity types"
     (?tag))
```

Figure 2.10: Implementation of Constraint 11 in EER-Tutor

EER-Tutor has been available on the Addison-Wesley's "Database Place" Web portal (www.aw-bc.com/databaseplace) providing learning opportunities for the international student community since 2004. It is not possible to measure the performance on pre- and post-test for this community due to the contract with Addition-Wesley. However, their learning processes were analysed in terms of how they learn constraints and compared with the student population at the University of Canterbury. The analysis revealed that both populations learn the constraints in a similar manner in terms of the initial frequency of making a mistake (i.e. violating a constraint) and rate of learning constraints.

NORMIT: NORMIT (Mitrovic et al., 2007) provides a practice environment to learn data normalization through problem solving. Data normalization is the process of refining relational database schemas in order to ensure that all tables are of high quality using a deterministic algorithm (Elmasri & Navathe, 2010). An overview of the procedure to be followed in NORMIT is given in Figure 2.11. (More details of this procedure are given in Section 3.1.3).

The student will only see the web page required to solve the current step. The student may submit the solution to the current task at any time and request feedback. The screenshot in Figure 2.12 represents the NORMIT's page corresponding to the first task, finding the candidate keys. In this situation (Figure 2.12), the student has incorrectly specified A as the candidate key, and asked for feedback, which is provided in the right pane. At any point during the session, the student can

change the problem, review the history of the session, examine the student model or ask for help on the current task. The system currently contains 50 problems and new problems can be added easily.

1. Identify candidate keys
2. Find the closure of a given set of attributes
3. Identify prime attributes
4. Simplify functional dependencies (FD), using the decomposition rule, if necessary
5. Determine the normal forms for the given relation. In this task, the student needs to determine whether the relation is in 1NF, 2NF, 3NF or BCNF. In this step, the expected answer is limited to Yes or No responses.
6. If the student indicated that the relation is not in 2NF, he/she expected to identify FDs that violate that form (i.e. partial FDs).
7. If the student indicated that the relation is not in 3NF, he/she needs to identify FDs that violate that form (i.e. transitive FDs).
8. If the student specified that the relation is not in BCNF, he/she will be asked to identify FDs that violate that form (i.e. FDs which do not contain a superkey on their left hand sides (LHS).
9. For relations that are not in BCNF, reduce LHS of FDs. This task checks whether some of the attributes on the LHS can be dropped while still having a valid functional dependency.
10. Find a minimal cover, which is a minimal set of FDs that still capture all the necessary information. In this task, the students need to apply the algorithm for checking whether a FD is redundant (and therefore can be dropped from the minimal cover) or not.
11. Decompose the table by using the minimal cover.

Figure 2.11: Problem-solving procedure adopted by NORMIT

The knowledge base of NORMIT is represented as a set of 82 (problem-independent) constraints (Mitrovic, Mathews, & Holland, 2012). Each constraint is relevant for a particular task of the procedure. Some constraints are purely syntactic, and check the structure of the solution, while others compare the student's solution to the ideal solution (generated by the problem solver).

When a solution is submitted, the system evaluates it and offers feedback. The first submission receives only general feedback, specifying whether the solution is correct or not. If there are errors in the solution, the incorrect parts of the solution are shown in red. In Figure 2.12, for example, the student has specified A as the key of the given relation, which is incorrect. On the second submission, the system provides a general description of the error, specifying general domain principles that have been violated. The third submission launches a more detailed message, providing a hint as to how the student should change the solution. It is possible to get a hint for every error in a student solution. The correct solution is provided only by request.

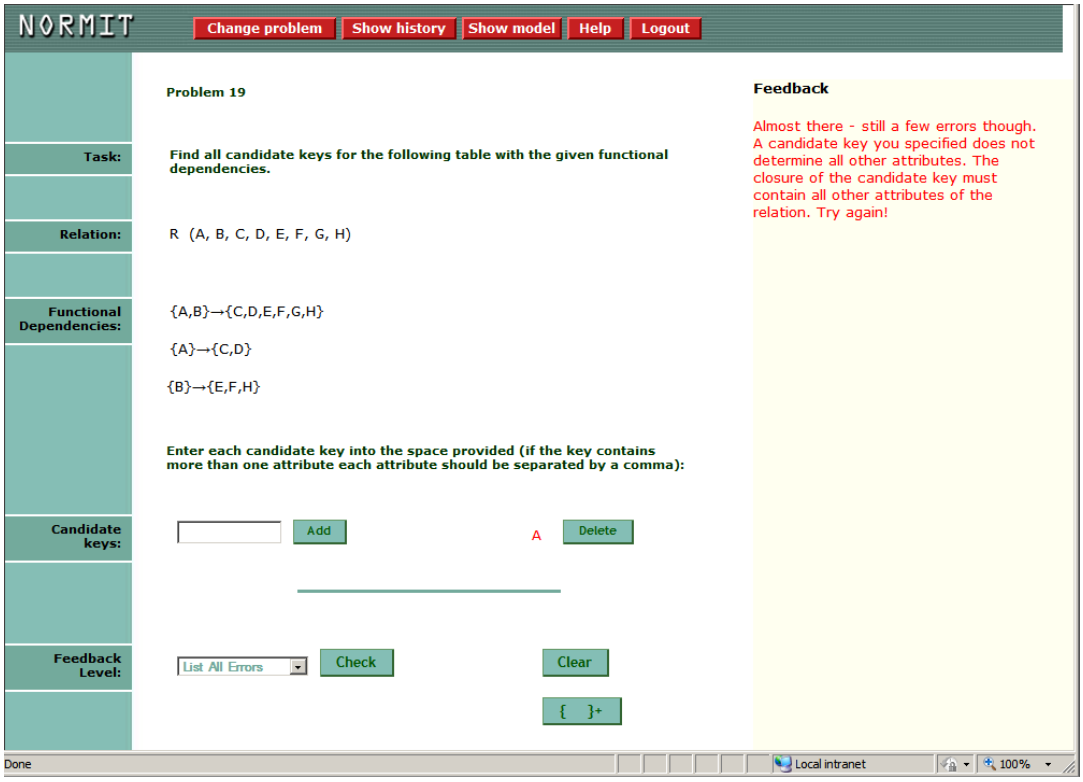


Figure 2.12: Interface of NORMIT

The short-term student model consists of a list of violated and a list of satisfied constraints for the current attempt. The long-term model records the history of usage for each constraint. This information is used to select problems of appropriate complexity for the student, and generate feedback.

2.3 Dialogue-based Systems

There have been a number of dialogue-based systems developed for a variety of domains. Some systems facilitate learner interactions via speech or typed responses. Our focus is on dialogue-based systems that interact with the learner via typed responses.

We define two dimensions to classify the dialogue-based systems: (i) main activity supported by the system; (ii) restrictiveness of the natural language supported by the system. The main activity supported by existing dialogue-based systems could be either (i) discussions to help students learn the domain knowledge; or (ii) problem solving. In the latter case, dialogues are used to provide additional learning support. The second dimension classifies systems based on whether the student responds in free-form natural language or by selecting from a list of pre-defined options.

The following subsections describe the tutorial aspects of a several significant systems that have been in use since 2000. We first focus on systems that provide problem solving as the main activity and use dialogues as additional learning support. We present CIRSCIM-Tutor, Atlas-Andes, Geometry Explanation Tutor, Ms. Lindquist and NORMIT-SE.

Then we focus on systems that involve students in discussions: AutoTutor, Why2-Atlas and Why2-AutoTutor. Section 2.4 presents KERMIT-SE, a problem solving environment for conceptual database design which uses dialogues as additional support. As this PhD project is based on KERMIT-SE, we discuss this system in a separate section. The focus of our research is on systems which use dialogues as additional support for problem solving, the main teaching activity.

Recently there have been other types of research carried out on dialogue-based systems. For instance, researchers are increasingly interested in investigating the effect of a student's affective states on learning in dialogue-based systems. Some of the affective states that researchers focus on are disengagement (Forbes-Riley & Litman, 2011) and confusion (Lehman et al., 2011). There is also growing interest within the ITS community to investigate how Machine Learning techniques could be used to further improve dialogue-based systems. Some researchers attempt to use machine learning techniques to detect learners' affective states from physiological signals such as heart activity, respiration, facial muscle activity and skin conductivity, while interacting with

a dialogue-based system (Hussain, AlZoubi, Calvo, & D'Mello, 2011). Other researchers focus on using machine learning techniques to improve the selection of dialogue strategies derived from studies of human tutorial dialogues (M. Chi, VanLehn, Litman, & Jordan, 2011a; M. Chi, VanLehn, Litman, & Jordan, 2011b). This kind of research is out of scope for the research discussed here.

2.3.1 CIRSCIM-Tutor

CIRCSIM-Tutor (Evans & Michael, 2006) assists students to learn about cardiovascular physiology relating to the regulation of blood pressure, using natural language dialogues. The tutoring strategies used to remediate student errors are a simulation of the tutoring carried out by two experienced human tutors. Students have the freedom to select a procedure that describes a perturbation of the cardiovascular system (shown on the upper right window of Figure 2.13). Then the system requests the students to predict the qualitative changes (increase, decrease, or no change) that will occur in seven cardiovascular parameters during the three time periods of the response: the Direct Response (DR) to the disturbance, the Reflex Response (RR), and the new, final Steady State (SS) (bottom right portion of Figure 2.13). Student input is limited +/-/0 to indicate the increase, decrease and unchanged statuses respectively in the variable that they currently focus on.

The ITS waits until the student has finished making predictions for the current stage (one whole column in the prediction table in Figure 2.13), then it compares the student's answers with the correct answers and marks the errors with a diagonal line in red. Then the ITS begins a natural language tutoring session to assist the student correct those errors. Within each stage, the variables are discussed in the sequence they are encountered in the solution of the problem. However there is one exception to the timing of the interventions. The system would intervene if the student starts the predictions with the wrong variable. The hint given by the system indicates the physical location in the human physiology system where the first change occurs as a result of the perturbation. The goal of this hint is to help the student to focus on the correct parameter to start predictions. This early intervention aims to avoid a large number of wrong predictions that occur as a result of starting predictions with the incorrect parameter.

The dialogues facilitated by CIRCSIM-Tutor are known as *Directed Line of Reasoning* (DLR) (Evans & Michael, 2006). A DLR is a multi-turn dialogue sequence in which the tutor helps the student reason about the problem with a series of questions, prompts and hints. This approach is often used to deliver explanations, summaries and remedies for misconceptions.

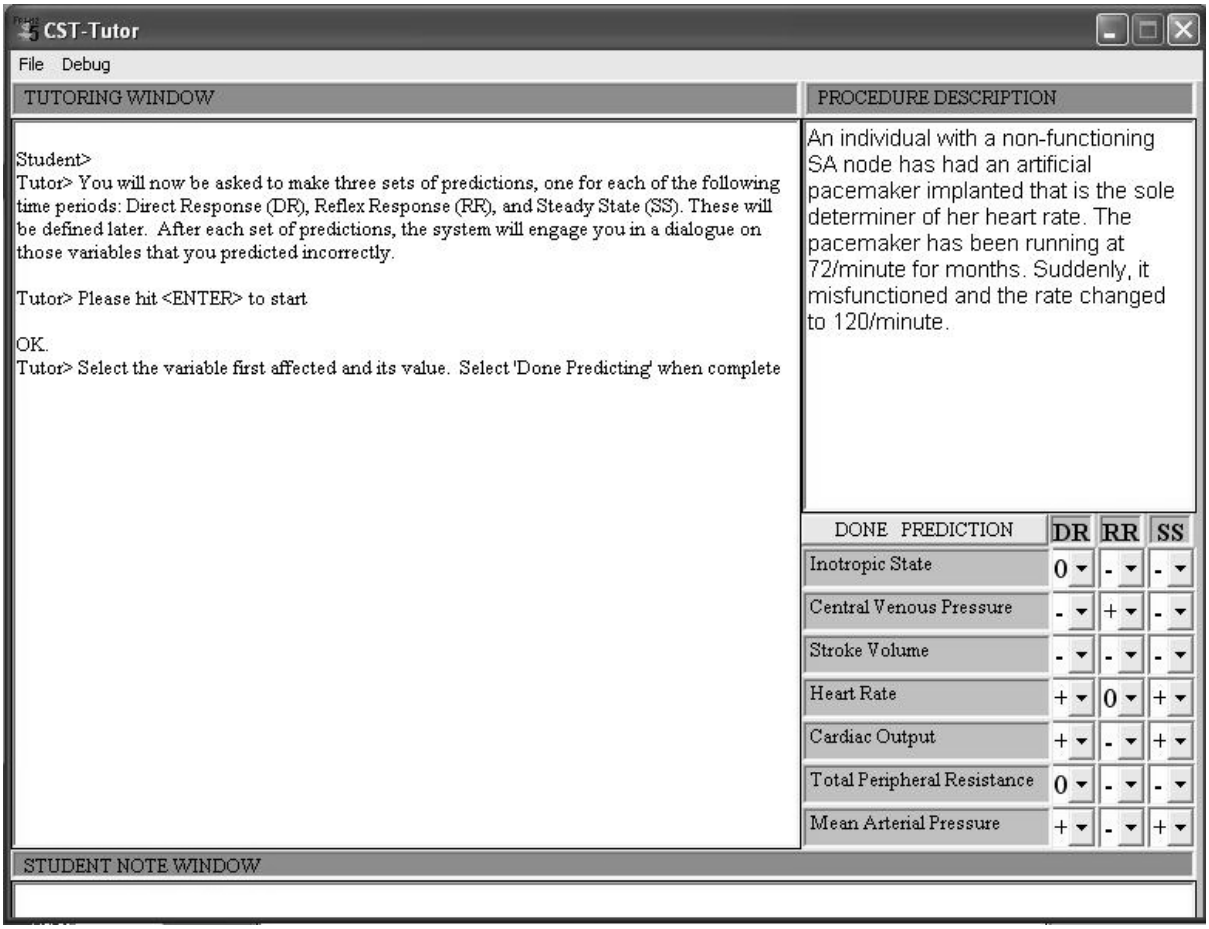


Figure 2.13: Interface of CIRSCIM-Tutor, from (Michael, Rovick, Glass, Yujian, & Evens, 2003)

The tutorial interactions consists of a sequence of tutoring dialogues focusing on incorrectly predicted variables, with an occasional additional topic, such as a summary, that is not immediately triggered by a student error. If all predictions are correct, the tutors generally ask a question or two to assess the student's understanding.

The system is capable of adapting the remedial dialogue to the student's learning needs as evidenced by his/her predictions and responses to the dialogue. The system can recognise a number of different types of student answers: partially correct but missing some essential

information, partially correct and partially wrong, *near-miss* (close that are not fully right), *grain of truth* answers (contains a correct concept that the tutor can use as the basis for a productive tutoring interaction), "I don't know" answers and totally wrong answers (Evans & Michael, 2006). This categorization enables a much wider range of responses. For instance, for a partially correct answer, the system acknowledges the correct part and provides a hint for the other part. For a student answer identified as a near miss, the system attempts to bridge the gap between the student answer and the correct answer. The response plans are dynamically generated from rules, but the system has some pre-specified plans to deal with serious misconceptions.

The effectiveness of CIRSCIM-Tutor was evaluated in several studies. The last study that was conducted in November 2002 involved version 2.9 of CIRSCIM-Tutor. The experimental group learnt with CIRSCIM-Tutor while the control group read a specially edited chapter on baroreceptor reflex (Evans & Michael, 2006). Both groups took a pre-test the weekend prior to the scheduled CIRSCIM-Tutor laboratory. The control group sat the post-test after reading the chapter whereas the experimental group did the same after interacting with the system for 1 hour in a scheduled CIRSCIM-Tutor laboratory. The control group consisted of 33 students whereas the experimental group had 40. At the end of study, the experimental group filled out a survey.

The pre and post-tests had 3 parts: (a) focused on the individual relationships between *n* cardiovascular variables making up the baroreceptor reflex, (b) a baroreceptor reflex problem to be solved using a prediction table and (c) a set of multiple choice questions posed in a clinical setting that required the application of the understanding of the cardiovascular system.

The results revealed that the CIRSCIM-Tutor was more effective than reading a text in teaching to predict the behaviour of the baroreceptor reflex. Even though the system was capable of assisting students to acquire knowledge of the relevant cardiovascular relationships, it was not as effective as reading the text. However the system was not specifically designed to teach such knowledge. The survey responses indicated that students liked the system and that they felt it was useful in acquiring the targeted knowledge.

2.3.2 Atlas-Andes

Atlas-Andes is the product of integrating the Andes physics tutoring system (Gertner & VanLehn, 2000; VanLehn et al., 2010) with the Atlas tutorial dialogue system (Freedman, Rose, Ringenberg,

& VanLehn, 2000). Andes is a model-tracing tutor that presents quantitative physics problems to students. Each problem-solving step entered by a student is highlighted in either red or green to indicate the accuracy of that step. ATLAS enhances the learning experience of ANDES by leading students through directed lines of reasoning to teach conceptual physics knowledge.

The main objective of dialogues provided by the system is to facilitate knowledge construction; hence, the dialogues are known as *Knowledge Construction Dialogues* (KCDs). These dialogues are influenced by CIRSCIM Tutor's directed lines of reasoning (VanLehn et al., 2007). They are designed to provide a solid foundation in conceptual physics, promote deep learning and enable students to develop meaningful problem-solving strategies.

In the fully integrated version of Atlas-Andes, KCDs are linked into Andes via the Conceptual Helper, replacing the original non-interactive mini-lessons (Rosé, Jordan, et al., 2001). Thus, KCDs provide unsolicited assistance to students in the first instance when they show evidence through incorrect problem-solving actions. The Conceptual Helper uses the Andes' solution graph to identify student errors. When an incorrect GUI action by a student occurs, Conceptual Helper attempts to find the correct action that the student was trying to accomplish. This correct action is determined by finding the closest action to the student's incorrect action in the graph that has yet to be performed. It then checks the existence of KCDs for any rule higher up in the graph that derives the matching step. If multiple KCDs exist that the student has not gone through since the beginning of his/her current session with Andes, then the KCD that is closest to the matching step is launched. Control is then passed to Atlas, and students are prevented from solving problems with Andes, until they have gone through the KCD or dismiss it.

A KCD starts with an introduction to the main tutorial objective and a scenario that will be used for illustrating a concept. Familiar scenarios are used to make the dialogues accessible to the broadest possible audience. The scenario is used to assist a student to draw out the intended conceptual knowledge by providing opportunities to make observations and predictions. A summary of the main point is given in the final turn of the dialogue.

An actual interaction that occurred between a student and the system is given in Figure 2.14. The objective of this KCD is to lead the student to understand the differences between speed and velocity, first in terms of their definitions and then in terms of how they are calculated. The first tutor turn (Tutor1 in Figure 2.14) describes the tutor's argument briefly for this KCD. It also aims

to bridge the gap between what the student may already know and what the student should learn. In this scenario it is the distinction between the speed and velocity. Then the tutor introduces the scenario and ends with a question about representing the displacement vector. All the KCDs follow this basic format at a high-level.

Tutor1:	Let's think about the difference between speed and velocity. A closely related distinction is that of the difference between distance travelled and displacement from the origin. Take as an example a bumblebee flying from point A to point B by means of a curvy path. If you draw a vector from point A to point B, you will have drawn the bee's displacement vector. What does the displacement vector represent?
Student1:	The distance travelled by the bee from its starting point to its ending point.
Tutor2:	*The displacement vector only represents the net change in position accomplished by the bee's motion. So the displacement vector does not tell you how far the bee travelled. It only tells you the net change in the bee's position. If you want to know the actual distance travelled by the bee, what entity do you have to measure the length of?*
Student2:	*The bee's path.*
Tutor3:	The bee's displacement vector is obviously a vector. Is the actual distance travelled by the bee a vector or a scalar?
Student3:	Scalar.
Tutor4:	So distance travelled is the scalar counterpart to displacement, which is a vector quantity. Now, can you tell me what is the difference between speed and velocity?
Student4:	Velocity is a vector composed of speed and direction.
Tutor5:	Speed is a scalar. So when you measure the bee's speed, do you want to use the distance travelled, which is a scalar, or the displacement, which is a vector?
Student5:	Distance travelled.
Tutor6:	So the equation for speed is the length of the path travelled by the body divided by duration of the motion, regardless of how curvy the path is, even if the path leads back to where the body started from.

Figure 2.14: A real example interaction between a student and Atlas, from (Rosé, Jordan, et al., 2001)

To respond to the tutor's question in first tutor turn, the student can type a response in a text box in natural language. The student may also avoid answering the question, by clicking on Continue. In both these cases, the system will engage the student in a remediation sub-dialogue designed to lead student towards the correct answer for the current question. Sub-dialogue selection is based on the content of the student responses as incorrect responses providing evidence of underlying misconceptions need to be handled differently to responses that imply knowledge deficits. In this example, the sub-dialogue explains the difference between distance and

displacement (Tutor2 in Figure 2.14). At the end of a sub-dialogue a check question is given to ensure that the student has understood the explanation. The student's correct answer (Student2 in Figure 2.14) provides evidence of successful integration of tutor's explanation into student's understanding.

Development of each of the fifty-five directed-lines of reasoning starts with a main line of reasoning consisting of a series of tutorial goals. Each goal is realised as a question accompanied by a short explanation, whenever possible. One or more expected answers are specified for each question. Each question also has a catch-all *anything else* case to ensure all possible students' answers can be dealt with. The expected answers together with the anything else case form a set of answer categories. A student's answer is categorised into one of the answer categories based on the defined grammar. This categorisation determines the subsequent tutor turns.

A set of remediation goals is developed for each expected wrong or partial student answer as well as the anything else case. Similar to the main line of reasoning, each remediation goal is associated with one or more lines of reasoning, each consisting of a sequence of tutorial goals. There are different types of remediation lines of reasoning. In simple cases, it can be a short explanation to assist the student to repair a detected misconception or a missing piece of information. In other cases, a more specific or simpler version of the previous question is presented to elicit the correct answer from the student. In situations where the missing or faulty knowledge can be further decomposed into a sequence of knowledge components, a multi-step directed line of reasoning known as a sub-dialogue is used. When a sub-dialogue is complete, the tutor proceeds with the main-line reasoning. The recursive, hierarchical structure of Atlas-Andes enables dialogues to be adaptive to each student's needs.

Studies that were conducted to evaluate the effectiveness of the KCDs against non-interactive minilessons revealed mixed results (Siler, Rosé, Frost, VanLehn, & Koehler, 2002). However, the last study demonstrated a trend in favour of KCDs with students who had no prior experience with college level physics courses (Rose, Bhembe, Siler, Srivastava, & VanLehn, 2003). This study compared student learning of basic physics concepts from KCDS to students learning from minilessons, non-interactive lessons that contain all of the same information provided by the main-line of reasoning from the corresponding KCD.

Students were given a pre-test consisting of 22 multiple-choice questions that focused on the target concepts taught by the KCDs or minilessons. Since the participants had no prior experience with college level physics courses, they were asked to read a six-page document summarizing the conceptual physics topics that were covered in the KCDs and minilessons used in the study. Students were assigned to either the KCD condition or the miniliesson condition based on their pre-test score. The pre-test score was used to balance the conditions as much as possible. In the KCD condition, students participated in 10 KCDs covering 10 topics including vector components, speed versus velocity, computing average velocity and computing average acceleration. In the minilesson condition, students read minilessons covering the main-lines of reasoning from their corresponding KCDs. In both conditions, after each KCD/minilesson, students were expected to provide an overview of the KCD or minilesson using a few sentences. In the KCD condition, students were prompted to elaborate their summaries once more after their initial summary. After students completed the KCDs/minilessons, they took a post-test, identical to the pre-test.

Sixteen students participated in the Minilesson condition and 19 in the KCD condition. Topic coverage was controlled but not for time on task. Students in the minilesson condition spent on average about half an hour to read all 10 minilessons, while their peers spent on average about 1 hour to go through all of the corresponding KCDs. Thus students required twice much time to engage in a KCD about a topic than to read a minilesson about that topic. There was no significant correlation between time on task and learning either within condition or over the entire population. The analysis was narrowed down to 14 pairs of participants from both conditions who had identical pre-test scores. For this subset of the population, participants who engaged in KCDs learnt significantly more than their peers who went through minilessons.

These KCDs discuss a domain concept using a new scenario (i.e. asking you to suppose that you were holding a rock in your hand or travelling in an elevator, when the original situation was about a block on an inclined plane). Having to be familiar with the new scenario can be viewed as additional work by some students, even though the dialogues are selected to cater for specific needs for each student.

Using Conceptual Helper to initiate KCDs prevents initiating them for certain types of errors as they are never identified by the Helper. For instance, the Conceptual helper cannot identify equation errors as they are identified by Andes only on demand. As initiating a KCD depends on

matching an incorrect action to a correct one in the solution graph, it is not possible to provide a KCD to assist a student on what to do next.

2.3.3 Geometry Explanation Tutor

The Geometry Explanation Tutor (Aleven, Popescu, Ogan & Koedinger, 2003) is the result of adding dialogue capabilities to the Geometry Cognitive Tutor (Anderson et al., 1996), which addressed the current geometry curriculum in high schools in the United States. The focus of the Geometry Explanation Tutor is on the Angles Unit which deals with the properties of angles in various kinds of diagrams. Students are presented with a diagram with a set of known angle measures, and are expected to find some unknown angle measures. Students are expected to explain their steps using geometry definitions and theorems. A previous version of this tutor expects the students to either type the name of the theorem or select it from a glossary of geometry knowledge (Aleven & Koedinger, 2002). The glossary listed the relevant geometry theorems and definitions and provided further information about each rule on demand. In the version with menu options, a typed explanation which seems to express the correct idea is considered correct even though it was not mathematically precise. In the later version, the Geometry Explanation Tutor engages students in natural language dialogue and guides them to explain their steps in their own words and produce mathematically precise explanations.

The Geometry Explanation Tutor uses a knowledge-based Natural Language Understanding (NLU) component and a simple dialogue management algorithm to assess and respond to student explanations. Each student input is assumed to be an attempt at stating an explanation and is processed in three steps.

First, the system parses the student's explanation, using the LCFLEX left-corner chart parser (Rosé & Lavie, 1999) and creates a semantic representation of the explanation, implemented in the Loom term description logic system (MacGregor, 1991).

Next, the semantic representation is classified based on an "Explanation Hierarchy", a fine-grained set of approximately 200 explanation categories. The categories represent students' incomplete or incorrect explanations commonly observed. The set of categories was developed by analysing several corpora of student explanations. Each category is associated with a sequence of messages that will be presented as feedback for an explanation classified under that category.

Finally, feedback to be presented is decided based on (a) the set of categories under which the explanation was classified and (b) the set of geometry rules that can be used to justify the current step. The rules applicable for the current step are identified by running the system's cognitive model for geometry problem solving.

The system produces feedback in the following manner. First, if the explanation categories indicate that the student explanation contains any geometric concepts that are not relevant to the correct geometry rules, the tutor indicates the concept(s) that need not be mentioned. Step 2 of the second dialogue in Figure 2.15 illustrates this scenario. If the explanation is totally irrelevant to any of the correct rules, the student is told that he/she is focusing on the wrong rule.

Second, if the student input is classified as a complete and correct explanation of the correct geometry rule justifying the current problem-solving step, the tutor accepts the explanation by means of implicit feedback. This scenario is illustrated by the last step of both dialogues in Figure 2.15.

1	**Student:**	the bottom angles are the same
2.	**Tutor:**	You are getting there, but you need to be more precise. Are base angles always congruent?
3.	**Student:**	the bottom angles are congruent
4	**Tutor:**	Base angles in what type of geometric figure are congruent?
5.	**Student:**	the bottom angles in an isosceles triangle are congruent
6.	**Tutor:**	[accepts as correct and complete explanation by means of implicit feedback]
1.	**Student:**	three angles in an isosceles triangle sum to 180 degrees
2.	**Tutor:**	Your explanation is close to being perfect. Except for the following: your explanation should not be specifically about isosceles triangles. Better delete that from your explanation.
3.	**Student:**	the sum of the three angles in all triangles is 180 degrees
4.	**Tutor:**	[accepts as correct and complete explanation by means of implicit feedback]

Figure 2.15:Two dialogues that students had with the Geometry Explanation Tutor, focused on the Isosceles Triangle Theorem and the Triangle Sum theorem (Aleven et al., 2003)

Finally, if the explanation is an incomplete statement of a correct geometry rule, the feedback focuses on hinting at or indicating what is missing. This is done by selecting the category (from the set of categories under which the student explanation was classified) closest to the one that represents a complete and a correct statement of one of the correct geometry rules. Then the system presents the first one from the series of feedback messages associated with that category. This case is illustrated in step 2 of the first dialogue in Figure 2.15. If the subsequent attempts do not indicate

an improvement of the explanation (i.e. the explanation is categorised under the same set of categories), the student receives the next feedback message in the sequence that provides more specific feedback. The first dialogue in Figure 2.15 illustrates this scenario. The student changed the explanation from "the bottom angles are the same" (step 1) to "the bottom angles are congruent" (step 3) to "the bottom angles are congruent". As both these explanations are categorised as having the same meaning, the next more specific feedback message attached to the chosen category (step 4) is presented.

A classroom study was conducted to test the hypothesis that students will gain a deeper understanding when they explain their problem-solving steps in their own words, as compared to explaining using a menu (Aleven, Popescu, et al., 2003) . The study took place within the context of a course based on the Integrated Mathematics Curriculum, which includes concepts both from algebra and geometry. It was conducted during three class periods, all taught by the same teacher. All students were honours students: they were the most gifted and diligent students within the given age group and school.

The students were assigned to two conditions, a "Dialogue" condition and a "Menu" condition, at the beginning of the study. Two entire classes were assigned to one of the conditions. The students in the third class were assigned randomly to one of these conditions.

Prior to the system interactions, the teacher and students covered the textbook chapter on proofs, which involves many of the geometry theorems that are covered in the system's Angles unit. Then students participated in an in-class, paper-and-pencil pre-test. During the same session, students watched a demonstration of the Geometry Explanation Tutor. All participants interacted with the system for four 40-minute sessions. In the final session, all students took a paper-and-pencil post-test.

The students in the Dialogue condition were asked to explain their reasoning steps in a (restricted kind of) dialogue while interacting with the Geometry Explanation Tutor. The students in the Menu condition explained their steps by specifying the name of a geometry definition or theorem. i.e. they used the previous version of Geometry Explanation Tutor, mentioned above. They could either type the name or select it from an on-line glossary of geometry knowledge, which listed the relevant geometry theorems and definitions. The glossary was available freely to

all the students in the study, but it functioned as a menu only for the students in the Menu condition. The two tutor versions were the same in all other respects.

Both pre-test and post-test included regular "Numeric Answer" and "Explanation" questions similar to the problems that students had encountered while interacting with the system. The tests also included two types of transfer items to assess the improvements in students' understanding. In some of these test items students were expected to determine whether there was enough information to find a particular unknown quantity. Items that involved a quantity whose value could not be uniquely determined are called "Not Enough Info" items. On the other hand, items that had a quantity whose value could be determined with the given information, were grouped with the Numeric Answer items. Some questions termed as "Verbal" items required students to determine the accuracy of a given general statement and to correct it, if applicable.

Of the 71 students, 62 completed the pre-test and post-test. The analysis focused on 46 students who worked on the tutor for at least 80 minutes, excluding the idle time. Even though the 80 minutes time threshold may seem somewhat arbitrary, Aleven and colleagues (2004) noted that results were not sensitive to the threshold. Dialogue condition included 21 students and Menu condition, 25 students. The results revealed that the participants who explained problem-solving steps by engaging in a dialogue with the system did not learn better overall than their peers who explained steps using a menu. This might be due the high pre-test scores. However the participants perform significantly better on the explanation questions. There was no significant difference in the improvement between the groups for other types of questions.

One of the limitations of these dialogues is the expectation to modify the complete explanation even when just a single word is missing. For instance, when the student justifies a step by saying "The angles in an isosceles triangle are equal" and the tutor responds with "Are all angles in an isosceles triangle equal?". It is not possible to say "No, it's just the base angles". Instead, the student is expected to modify the complete explanation to say "The base angles in an isosceles triangle are equal." This is due to system's inability to break down the knowledge construction process through new non-rhetorical questions and multi-step plans.

2.3.4 Ms. Lindquist

Ms. Lindquist (Heffernan et al., 2008) was developed to help students using dialogues to develop the skill symbolization, i.e. writing algebraic expressions for word problems. It models both student behaviour and tutorial behaviour by combining a cognitive model of student behavior with a tutorial model of strategies observed in human tutors. The cognitive student model has a set of production rules that models the problem-solving skills required to write algebraic expressions. The tutorial model is based on the observations of an experienced human tutor and thus includes tutorial strategies that were observed to be effective for this instructional task.

Tutorial planning is done through a tutorial agenda, a data structure that behaves like a stack, and is used to keep track of the current focus. It includes questions that the student has already been asked but are still awaiting a correct response, as well as questions that the tutor plans to ask but has not yet done so. The question at the top of the agenda represents the current question that the student was just asked. When a question is answered correctly, it is removed from the agenda and remaining questions are posed to the student.

Sometimes student responses may include more information than what was expected by the system. In such cases, Ms. Lindquist can identify that the response actually answers one or more questions lower down in the tutorial agenda and removes these questions that have been answered. For example if the expected answer to the current question was “m/s” but the student answer was “b + m/s”. Rather than indicating that it is an incorrect response for the current question, the system accepts it as the correct answer for the question asking for “b+m/s”. All the questions between the current question and the one that expects “b+m/s” as its answer are removed from the agenda.

If the student response is incorrect the tutors says “No” and then tries to add some positive feedback for any correct aspect of his/her answer before passing the control to the dynamic scaffolding procedure. The purpose of this procedure is to produce a plan to address the student error in the given context. Dynamic scaffolding is based on human tutors’ tendency to ask questions related to incorrect aspects of a student’s answer. Localizing the error in this way communicates valuable information to the student by focusing the student’s attention on a single aspect during a complex problem-solving process. When the aspect of the student answer to focus

upon has been determined, the next step is to decide the most pedagogically effective tutorial strategy for the given context. Selection of the best tutorial response is based on simple heuristics called selection rules. Some examples of these rules are discussed below. The questions associated with the selected strategy are placed in the agenda to be presented to the student.

The tutorial strategies in Ms. Lindquist are categorized as Knowledge Remediation Dialogues (KRD) and Knowledge Construction Dialogues (KCD). KCDs are similar to the knowledge construction dialogues in Atlas-Andes. Both KCDs and KRDs invoke multi-step plans to discuss errors. KRDs are applicable only for specific types of errors such as missing parentheses. In contrast, KCDs have a broader applicability than KRDs. Due to the limited applicability of KRDs, the system has only one KRD which deals with errors of omission. i.e. when only a part of the correct algebraic expression is articulated. For example, if the correct expression is 800 - 40m but the student response was 40*m, the system prompts the student to indicate what 40*m represents. When the student correctly responds, he/she asked to make another attempt to articulate the entire expression again.

On the other hand, four KCDs have been developed: Concrete Articulation Strategy, Inducted Variable Strategy, Explain in English Strategy, and Convert the Problem into an Example to Explain Strategy. Each strategy is a sequence of question types to be asked from the student. For instance, Concrete Articulation Strategy consists of the sequence: Q_compute (find a numerical answer), Q_explain (write a symbolization for a given arithmetic quantity, and Q_generalize (use the result of Q explain abstractly).

Selection of the best tutorial response from a set of possible different responses for a given context is based on simple heuristics called selection rules. For instance one selection rule orders the possible responses in this sequence: KRD, Buggy-Message, KCD, and finally hint. The reason for this sequence is to provide a response that considers as much context as possible (KRD & Buggy Message). Another heuristic is to use a tutorial strategy (KRD or KCD) before using a buggy message or hint. This is due to the higher priority assigned by the system to asking questions than providing a hint. Buggy-Messages and hints are similar to those available in the traditional model tracing tutors (Corbett et al., 1998). Hints and buggy-messages are examples of single-turn strategies in contrast to the KCDs and KRDs that consist of multiple steps.

A hypothetical dialogue that contains an embedded sub-dialogue is presented in Figure 2.16. The student response of "2+5" is considered a partially correct as '5' was not articulated properly. An embedded sub-dialogue is launched to assist the student to specify where the '5' came from (T4 in Figure 2.16). When the student managed to respond correctly, the tutor could proceed with the rest of the plan (indicated by saying "Now, let's try this question again." in T5).

T1: Hello. [Doing the "bike-trip" problem] Please write an expression for the total time the trip took. [Q_symb]

S1: help! [Student hit the help button]

T2: Let me ask you an easier question. Please compute the time actually on bikes if instead the speed was 20 miles per hour, the time of the break was 2 hours, and the distance ridden was 100 miles? [Q_compute]

S2: 7

T3: Correct. What math did you do to get that 7? [Q_articulate]

S3: 2+5

T4: Okay, but how did you get the 5? [Q_articulate again but this time for the 5.]

S4: 100/20

T5: Correct. Now, let's try this question again. What math did you do to get the 7? [Q_articulate]

S5: 2+100/20

T6: Correct. Now write your answer of '2+100/20' using the variables given in the problem (i.e. put in 'm', 'b' and 's')? [Q_generalize]

S6: b+m/s

T7: Correct.

Figure 2.16: A hypothetical dialogue in Ms.Lindquist (Heffernan et al., 2008)

The effectiveness of Ms. Lindquist was evaluated in several experiments (Heffernan & Croteau, 2004) that showed positive learning results for the ITS. One evaluation focused on 76 middle school students who used Ms. Lindquist as part of an assignment. The students in the experimental condition engaged in dialogues associated with the Concrete Articulation strategy when they needed help. The participants in the control condition were given the answer if they answered incorrectly and were moved on to the next problem. This experiment was controlled for time. The interaction between conditions and learning gain was statistically significant with an effect size of 0.56 standard deviations in favour of Ms. Lindquist, even though the students in the control group did significantly fewer problems than those in the experimental group.

2.3.6 NORMIT-SE

NORMIT-SE (Mitrovic, 2005) is the result of enhancing NORMIT (Section 2.2.3) to facilitate self-explanation using tutorial dialogues. In contrast to dialogue-based systems such as Geometry Explanation Tutor, NORMIT-SE expects an explanation for each action type that is performed for the first time. For the subsequent actions of the same type, explanation is required only if the action is performed incorrectly. This approach would reduce the burden on more able students (by not asking them to provide the same explanation every time an action is performed correctly), and also that the system would provide enough situations for students to develop and improve their explanation skills.

Students provide explanations by selecting one of the offered options. The order in which the options are given is random, to minimize guessing. For example, if the specified candidate key is incorrect, NORMIT-SE asks the following question:

This set of attributes is a candidate key because:

- *It is a minimal set of attributes*
- *Every value is unique*
- *It is a minimal set of attributes that determine all attributes in the table*
- *It determines the values of all other attributes*
- *All attributes are keys*
- *Its closure contains all attributes of the table*

The offered options are not strict definitions from the textbook, and the student needs to reason about them to select the correct one for the particular state of the problem. This approach does not rely on student's memory for his/her ability to select the correct explanation, but requires the student to re-examine his/her domain knowledge. Therefore, this kind of support requires recall and is comparable to generating explanations.

If the student's explanation is incorrect, he/she will be given another question, asking to define the underlying domain concept (i.e. candidate keys). An example of such a question is given below.

A candidate key is:

- *an attribute with unique values*
- *an attribute or a set of attributes that determines the values of all other attributes*
- *a minimal set of attributes that determine all other attributes in the table*
- *a set of attributes the closure of which contains all attributes of the table*
- *a minimal superkey*
- *a superkey*
- *a key other than the primary key*

In contrast to the first question, which was problem-specific, the second question focuses on domain concepts. If the student selects the correct option for a question, he/she will resume with problem solving. If the student's answer is incorrect, NORMIT will provide the correct definition of the concept.

An evaluation study was conducted to investigate the effect of explaining problem-solving steps on both procedural and conceptual knowledge (Mitrovic, 2005). The study involved 49 students enrolled in an introductory database course at the University of Canterbury. The control group used NORMIT while the experimental group used NORMIT-SE. Prior to the study students had four lectures and one tutorial on data normalization. The system was demonstrated at a lecture and was open to the students a day later. The pre-test was administered on-line at the beginning of the first session. The pre-test consisted of four multichoice questions with a maximum mark of 4. The students were free to use the system when and for how long they wanted. The post-test was administered as a part of the final examination for the course. This type of post-test was chosen as the study was not controlled, and this was the only way to ensure that each participant sits the post-test. The post-test was longer with a maximum of 29 marks. As a result, pre- and post-tests were not directly comparable. The students who explained their problem-solving steps learned constraints significantly faster than their peers who did not. There was no significant difference between the two conditions on the post-test performance, and it might be due to the short duration of their sessions interacting with the system. Furthermore, the analysis of the self-explanation behavior shows that students find problem-specific question (i.e. explaining their action in the context of the current problem state) more difficult than defining the underlying domain concepts.

The students' conceptual knowledge improved regularly during their interaction with NORMIT-SE.

2.3.7 AutoTutor

AutoTutor (Person et al., 2001; Graesser, Rus, D'Mello, & Jackson, 2008; Graesser, 2011) teaches topics in computer literacy such as Hardware, Operating Systems and the Internet by having a conversation with students. AutoTutor requires students to provide lengthy explanations for *How*, *Why* and *What-if* type of questions. This approach encourages students to articulate lengthier answers that exhibit deeper reasoning instead of short answers, which may lead to shallow knowledge. The natural language processing components of the tutor are based on Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997).

Several systems have been evolved from the original AutoTutor to cover different domains including biology (Graesser, D'Mello, & Person, 2009) (GuruTutor), research ethics (Hu & Graesser, 2004) (HURA Advisor), critical thinking in science (Millis et al., 2011), physics (Graesser, Franceschetti, Gholson, & Craig, 2011) and self-regulated learning (Azevedo, Johnson, Chauncey, & Burkett, 2010). Here we focus on some of the important details of the AutoTutor teaching computer literacy.

Figure 2.17 presents the interface of Why2-AutoTutor (VanLehn et al., 2007), an ITS similar to AutoTutor that teaches qualitative physics. The animated agent that acts as a dialogue partner with the student appears on the top-left corner (Figure 2.17). The agent delivers AutoTutor's dialogue moves with synthesized speech, intonations, facial expressions and gestures. The problem that the student receives is both spoken by AutoTutor and is printed at the top of the screen. Questions in the dialogue are generated systematically from a curriculum script, a module discussed below. The system interaction involves a multi-turn, mixed-initiative dialogue with the student. When the turns of both the learner and AutoTutor are considered, it takes 10 to 20 conversational turns to answer a single question or solve a problem from the curriculum script.

The curriculum script is a sequence of *topic formats*, each of which contains a *main focal question*, and a pre-specified *ideal complete answer*. The ideal complete answer consists of several sub-answers, called *aspects*. Each aspect has the following information: (i) a list of expected bad

answers corresponding to misconceptions that need correction (with splicing/correcting moves); (ii) lists of prompts and hints that can be used to encourage the learner to contribute information and (iii) elaboration and summary moves that can be used to provide the learner with additional or summarizing information. All these moves are pre-specified.

The tutoring session starts with a brief introduction and then a question from the curriculum script is presented to the student. AutoTutor evaluates the student's answer against all the aspects of the ideal complete answer, and the anticipated bad answers. The system provides immediate feedback based on the student's answer, and then executes dialogue moves that encourages the learner to contribute information until all answer aspects are sufficiently covered.

Figure 2.17: User interface of Why2-AutoTutor (VanLehn et al., 2007)

Selecting these dialogue moves is based on (i) the quality of the student's assertion in the preceding turn; (ii) global parameters like student ability, verbosity and initiative; and (iii) the extent to which the good answer aspects have already been covered. A set of fuzzy production rules determines the category of the dialogue move to be selected. The content of the dialogue move is computed by an algorithm that selects the next good answer aspect to focus on. Currently

12 dialogue moves are possible: Pump, Hint, Splice, Prompt, Prompt Response, Elaboration, Summary, and five forms of immediate short-feedback (positive, positive-neutral, neutral, negative-neutral, and negative).

Figure 2.18 shows a dialogue between a college student and AutoTutor. Prior to the first question in the Figure 2.18 (TUTOR-1), the student has attempted to answer 6 previous questions about the internet. Tutor-2 is an example of a Pump, which is used to elicit more information from the student. Prompts are used to encourage the learner to produce a single word as shown in Tutor-3 and Tutor-4. Assertions are given in Tutor-5 and Tutor-6.

TUTOR-1:	Alright, let's go on. You want to take photos and send them over the Internet. Consider this problem. What hardware do you need to take photos and send them over the Internet?
STUDENT-1:	First of all you need to have access to the internet.
TUTOR-2:	What else?
STUDENT-2:	a camera
TUTOR-3:	Yeah. For what type of connection do you need a network card?
STUDENT-3:	network card
TUTOR-4:	Okay. Now, if you have access to the Internet through a network card, then your connection is…
STUDENT-4:	I don't know.
TUTOR-5:	Is direct. Okay. A network card is needed if you have a direct connection to the Internet. Okay. How does the user get hooked up to the internet?
STUDENT-5:	by a plug.
TUTOR-6:	Once again, a network card is needed if you have a direct connection to the Internet. Okay, what does a modem do?
STUDENT-6:	connects network card and internet
TUTOR-7:	Okay. Try this. A modem is required when there is…
STUDENT-7:	a connection
TUTOR-8:	a dial up connection.
..	
TUTOR-30:	Let's review. To send your photos on the internet, you need either a digital camera or a regular camera to take the photos. If you use a regular camera, you need a scanner to scan them onto a computer disk. If you have a direct connection to the Internet, then you need a network card. A modem is needed if you have a dial up connection.

Figure 2.18: An Example Tutorial Dialogue between AutoTutor and a learner (Graesser, VanLehn, Rosé, Jordan, & Harter, 2001)

During the discussion of a topic, the system needs to keep track of both the good answer aspects that have been covered and the dialogue moves that have been generated. In AutoTutor 1.1, LSA Topic Coverage metric is used to decide whether each good answer aspect for a topic has been covered. LSA computes the extent to which the various tutor and student turns cover each good answer aspect associated with a particular topic. The Topic Coverage metric varies from 0 to

1 and gets updated for each good answer aspect with each tutor and student turn. When a pre-specified threshold is met or exceeded, then good answer aspect is considered to be covered. A topic is finished when all of aspects have coverage values that meet or exceed the threshold.

One of the other important decisions is to select which aspect to focus on next. Different versions of AutoTutor employ different strategies for this selection. AutoTutor 1.1 uses the idea of zone of proximal development (Vygotsky, 1978) to select the good answer aspect to focus on next. It selects the aspect that has the highest subthreshold coverage score. This selection mechanism attempts to build on the fringes of what the student knows. AutoTutor 2.0 uses two additional features: discourse coherence (selecting the aspect that is most similar to the previous one that was covered) and pivotal aspects (selecting an aspect that has the greatest content overlap). In addition, AutoTutor 2.0 uses discourse patterns that organize dialogue moves in terms of their progressive specificity. Hints are less specific than prompts, and prompts are less specific than elaborations. Thus, AutoTutor 2.0 cycles through a Hint-Prompt-Elaboration pattern until the student articulates an aspect. The other dialogue moves (e.g., short feedback and summaries) are controlled by the fuzzy production rules available in AutoTutor 1.1.

The system has undergone several evaluation studies. They reveal that students interacting with AutoTutor repeatedly learnt significantly more than students who study a text book for a similar amount of time (Person et al., 2001; Graesser et al., 2003; Graesser et al., 2004; Graesser et al., 2008; Graesser, 2011).

2.3.8 Why2-Atlas

The Why2-Atlas system (VanLehn et al., 2002; Makatchev, Hall, Jordan, Pappuswamy, & VanLehn, 2005; Jordan, Makatchev, Pappuswamy, VanLehn, & Albacete, 2006) teaches qualitative physics by analysing student explanations of simple mechanical phenomena. The system uses the knowledge construction dialogues in Atlas-Andes (Jordan, Ringenberg, & Hall, 2006). The system uses deep syntactic analysis and abductive theorem proving to convert the student's explanation to a proof. The proof formalizes the content of the essay as well as student's beliefs behind what is being constructed. A list of expectations and misconceptions are specified for every problem. Based on this list and the proof, the tutor is able to identify both misconceptions

and missing content from the explanation. Dialogues are used to assist students to remedy missing or misconceived beliefs.

Each error identified is queued as a tutorial goal to be discussed with the student. For an essay with multiple errors, the highest priority goal is selected from the queue. The priorities are defined as follows: discuss misconceptions before anything else, then fix self-contradictions, errors and incorrect assumptions and finally elicit missing mandatory points. It is common for students to miss several mandatory points. In such cases, selecting the pedagogically most suitable point to discuss is based on a hand-authored list for each problem.

Tutorial goals for remedying a misconception are associated with a specific remediation knowledge construction dialogue (KCD), whereas those for eliciting a mandatory point are associated with a specific elicitation KCD. Questions in the KCDs are designed to elicit short answers from students to make the process of matching expected responses to the actual ones simpler.

Figure 2.19 represents a KCD provided by Why2-Atlas. This KCD has four steps, each uses a question to elicit relevant reasoning (Lines 1, 3, 5, and 7). A summary is provided in the last tutor turn (Line 9). If the tutor fails to detect a correct answer to its question (Line 2), the tutor initiates a sub-dialogue (Lines a through c). Different wrong answers initiate different sub-dialogues and they could be nested (Line i).

1. Tutor: Although it is clear that the truck exerts a force on the car, does the car also exert a force on the truck?
2. Student: I don't think so.

    a. Tutor: Could the car possibly dent the truck?
    b. Student: Not likely.

        i. Tutor: The car may not always dent the truck, but if the car was heavy enough and hit a soft part of the truck, then it could dent the truck.

    c. Tutor: If the car can dent the truck, then it must be exerting a force on the truck, and sometimes the force is enough to dent the truck.

3. Tutor: So when the truck and car collide, they each exert a force on the other. Whenever that happens, namely, two objects exerting forces on each other, the pair of force is called what?
4. Student: Action–reaction.

5. Tutor: The forces are an action–reaction pair. What is the relationship between their magnitudes?
6. Student: Same magnitudes but opposite directions.

7. Tutor: What is the relationship between the directions of the force on the car and the force on the truck?
8. Student: Opposite.

9. Tutor: So the car exerts a force on the truck, and at the same time the truck exerts an equal and opposite reaction force on the car.

Figure 2.19: A knowledge-construction dialogue (KCD) in Why2-Atlas, from (VanLehn et al., 2007)

When a student's engagement with a KCD is complete, the student is asked to revise the essay and resubmit it. Sometimes a student may fail to correct an error even after the tutor discussed it. If that error was selected for discussion, a second KCD will be initiated, if available. Multiple KCDs are developed to discuss pedagogically significant points. If there are no KCDs to be initiated for repeated errors, then the tutor specifies the change explicitly by saying "It could be that you have what I'm looking for in mind, but I'm not just able to understand what you're saying. Let me show you what I'd say was the point that should be covered in your essay. <text>"

Among other studies involving Why2-Atlas, one compared learning gains with three other conditions: learning with Why2-AutoTutor, text-only condition (students studied the canned text, including the questions and the ideal answers, but did not write answers on their own) and canned text-remediation (in which students write an essay, read some text designed to remedy potential flaws and edited their essay) (Vanlehn et al., 2007; Chi, VanLehn, Litman, & Jordan, 2011). Only those students who interacted with tutoring systems received feedback on their essays. Even though the text used in the canned text-remediation condition addressed potential flaws, the same text was given to all the participants regardless of the flaws in their essays. This study involved novices who had not taken a college level physics course prior to the study. Both the learning material provided by the system and the text were redesigned for novices as the existing material were intended for intermediate students who had taken college level physics courses. Participants were asked to go through pre-training material, sit a pre-test, construct four essays or read text based on the condition and sit a post-test. Pre-training material consisted of seven lessons, one discussing each major physics principle selected for the study. Each lesson consisted of about two pages of text and one or more canned-text exercises. The pre-test contained 15 multiple choice and two essay questions. The post-test had 14 multiple choice, five fill-in-the-bank and six essay questions. Each fill-in-the-blank problem asked a top-level question that was similar in complexity to the essay questions, but provided a paragraph-long answer with blanks in key places. These questions were designed to assess students' ability to compose multi principle explanations with some scaffolding. Even though students in all four conditions learned the same amount, Why2-Atlas students scored significantly higher on the fill-in-the-blank question than the canned text remediation students.

2.3.9 Why2-AutoTutor

Why2-AutoTutor (VanLehn et al., 2007) was developed for the domain of qualitative physics, the same target domain as Why2-Atlas. In contrast to Why2-Atlas which expects the student to revise the essay after each KCD, Why2-AutoTutor tries to elicit the correct expectation from the student. If these attempted elicitations fail, the tutor asserts the expectation. If the emerging explanation indicates another error or a misconception, the tutor asks a question to verify the misconception and attempts to correct it.

When Why2-AutoTutor decides that all flaws in a student's beliefs about the problem have been remedied, three dialogue moves always occur. First, the tutor randomly selects one of the anticipated misconceptions for the problem and asks a diagnostic question. If the student's response is incorrect, the tutor corrects the misconception in a single turn and goes on. The purpose of the diagnostic questioning is to facilitate remedying misconceptions in case the LSA analysis is unable to detect misconceptions.

The second move is requesting the student to ask a question to encourage a mixed-initiative dialogue. Then the tutor attempts to answer the question by classifying it into one of two different categories. The last move is asking the student to enter a complete essay. This essay is not used to control the multi turn dialogue. Regardless of the quality of the final essay, the tutor presents the ideal answer. When the student finishes studying the ideal answer, the tutor moves to the next physics problem.

A study conducted to evaluate the effectiveness of Why2-AutoTutor indicated significant learning gains from pre-test to post-test when interacted with the system in comparison to a text-book and a no-instruction condition (VanLehn et al., 2007). Sixty-seven university students who were taking college level physics courses but not advanced physics courses participated in the study. Students were assigned randomly but unevenly to conditions: Why2-AutoTutor (N= 32), textbook (N= 16) and no instruction (N= 19). This is because at least 30 participants were needed to conduct correlational analyses. While the Why2-AutoTutor students received tutoring on their essays and were expected to revise their essay, students in the textbook condition were expected to only study the text (i.e. writing and revising essays was not required). The textbook was designed by selecting pages from Hewitt's (1987) Conceptual Physics textbook that covered the target

domain principles. Students in the no-tutoring condition simply took the pre-test in the first session and the post-test in the second session.

All students filled out a background questionnaire on their physics courses, took a pre-test, went through one of the three conditions, took a post-test, and completed an attitudinal questionnaire. Each of the pre-test and post-test consisted of four essay questions and fourty multiple choice problems. Essay questions were designed to address the same principles and misconceptions as the problems the students encountered during the construction of explanations and did not require any additional knowledge. Why2-AutoTutor students were expected to construct ten essays.

2.4 KERMIT-SE: Extending KERMIT to facilitate Self Explanation

KERMIT was extended with dialogue capabilities to facilitate self-explanation (SE), in my M.Sc. research. For a detailed discussion on how KERMIT was enhanced to support self-explanation see (Weerasinghe, 2003; Weerasinghe & Mitrovic, 2006). Some of the important details are discussed here. As mentioned in Section 2.3, KERMIT-SE provides a problem-solving environment and uses dialogues as additional learning support. User responses are limited to selecting the correct explanation from a pre-defined list. All the systems that facilitate self-explanation prior to KERMIT-SE prompt students to explain most of the problem-solving steps, requiring students to point out the definitions/theorems used. We believed this approach puts too much burden on able students. Therefore, our tutor, KERMIT-SE prompted for self-explanation only when the student violates a constraint, which indicates missing/erroneous knowledge or a slip. The tutor is thus able to customise self-explanation based on the student solution so that the knowledge construction is facilitated for students who have misconceptions or gaps in their knowledge without disrupting others (Bunt, Conati, & Muldner, 2004).

The model to facilitate SE consists of two parts: an error hierarchy and tutorial dialogues. The error hierarchy categorizes all error types in the domain of conceptual database design. At the lowest level, an error type is associated with one or more violated constraints, which form leaves of the hierarchy. The error types are then grouped into higher level categories. Self-explanation is facilitated through tutorial dialogues, one of which is developed for each error type. When there

are multiple errors in a student solution, the hierarchy is used to select the error most suitable for discussion and the corresponding dialogue is then initiated. In response to the initiated dialogue, learners are able to provide answers by selecting the correct option from a list. Each component is now described in detail.

Error hierarchy: Since a student solution can contain several errors (i.e. several constraints can be violated) in a single submission, the pedagogical module (PM) needs to decide on the pedagogically most suitable error to initiate the self-explanation process. Selecting the most suitable error to discuss is crucial as this helps students to build a comprehensive mental model of the instructional domain. The simplest approach is to use the order in which the constraints are specified in the constraint base. The constraints are ordered according to traditional ER modelling procedure, starting with entities first, then relationships and finally the attributes. This ordering alone is not sufficient to select an error to prompt for self-explanation, as most solutions violate more than one constraint. At the same time, some semantic constraints are not specific enough to guide self-explanation effectively. For instance, the constraint *A construct that should be a regular entity has been represented by another type* is violated when a regular entity is modelled either as a simple attribute or a weak entity. Different approaches are required in these two cases. Self-explanation in the first case should help the student to clarify the definitions of entities and attributes so that the student understands when to use them. In the latter case, the student should understand the differences between regular and weak entities, which will enable the error to be corrected and the correct design decisions made in future. Also, the pedagogical module should enable students to build a more comprehensive mental model of the domain knowledge by giving them an opportunity to learn basic concepts before complicated ones. For instance, students need to understand the reason for a certain component to be modelled as a regular entity before understanding the relationships that this entity participates in or attributes of that entity. Our approach to select the pedagogically most suitable error for discussion was based on an error hierarchy that categorizes all the errors in the conceptual database design. This hierarchy was used to introduce a high-level structure of constraints to the constraint base of KERMIT. The development of the hierarchy was done in consultation with a domain expert. We also used students' answers for an assignment in an introductory database course and students' responses to

pre- and post-tests of a previous evaluation study conducted for KERMIT (Suraweera & Mitrovic, 2004). The overall view of the error hierarchy is shown in Figure 2.20.

In CBM domain knowledge is represented as a set of constraints on correct solutions. These constraints identify the correct solutions from the space of all possible solutions. Constraints represent only declarative knowledge of the domain. Incorrect solutions can be identified as solutions that contravene the semantic and syntax rules of the domain. In other words an incorrect solution violates one or more constraints specified for a domain. Incorrect domain knowledge is much greater than correct knowledge and violates one or more constraints for that domain. The proposed error hierarchy divides the space of incorrect knowledge into syntax errors and semantic errors. Each one is further divided as presented in Figure 2.20.

ALL ERRORS
- Syntax Errors
  - Errors dealing with simple connections
    - 1
    - 2
    - 3
    - 4
    - 5
    - 6
    - 101
    - 7
    - 8
    - 9
    - 10
    - 21
- Semantic Errors
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Connecting an attribute to an incorrect construct
  - Errors dealing with cardinalities and participation

Figure 2.20: Overall view of the error hierarchy

Violated constraints (which are specified by the constraint numbers in Figures 2.20 and 2.21) for each type of error are represented as leaves of the hierarchy. Constraints for the nodes in Figure 2.20 are given in separate lines to indicate that each constraint deals with a specific type of error.

Our experience in teaching conceptual database design in a traditional classroom setting at the University of Canterbury indicates that it is easier for students to understand and correct syntax

errors than semantic ones. Therefore, the error hierarchy focuses on dealing with the syntax errors before the semantic ones. This is achieved by placing the node *Syntax errors*, which deals with all the syntax errors before the node *Semantic errors*, which deals with all the semantic errors (Figure 2.20).

The nodes in this hierarchy are ordered from the basic domain principles to more complicated ones so that a student can be guided systematically towards building and repairing his/her mental model of the domain. Therefore, when the hierarchy is traversed to find the first leaf that corresponds to one of the violated constraints, the simplest error for a given student solution can be found.

Syntax errors: The constraint base of KERMIT contains two types of constraints: syntax and semantic constraints. The syntax constraints focus on the syntactic accuracy of the student solution. On the other hand the semantic constraints deal with how a student solution matches a given scenario by comparing it to an ideal solution that a human teacher deemed as pedagogically the most suitable solution. Syntax constraints are of varying complexity. For instance, constraint 1 is violated when more than one ER construct is used to model a phrase of the problem statement (i.e. when the phrase "CHAPTER" is modelled by both a regular entity and a simple attribute), whereas constraint 45 is violated when a regular entity does not have a primary key attribute. Thus, constraint 1 deals with a relatively simpler error than constraint 45. As a result, it is sufficient to use a simple feedback measage when constraint 1 is violated. However it is better to initiate a longer dicussion for constraint 45 as it is more complex. The syntax constraints for which a simple feedback message is sufficient have been categorised under the node *Syntax errors* in the error hierarchy (Figure 2.20). The node *Semantic Errors* categorises both the remaining syntax constraints for which a detailed discussion is needed and the semantic constraints.

Semantic errors: Each of the semantic error types are further divided into sub error types as indicated in Figure 2.20. For instance, the node *Using an incorrect construct type* is divided into two child nodes: (i) *Using a completely different type of construct* and (ii) *Using a different variation of the correct construct*. In ER modelling, two types of design decisions need to be made. The first type of decision requires deciding whether a certain phrase in the problem statement can

be modelled as an entity, relationship or an attribute. The second type of decision focuses on deciding whether it is regular or weak in the case of an entity; regular or identifying in that of a relationship etc. The error types associated with the first step are classified under the node *Using a completely different type of construct*. The node *Using a different variation of the correct construct* deals with the errors related to the second step.

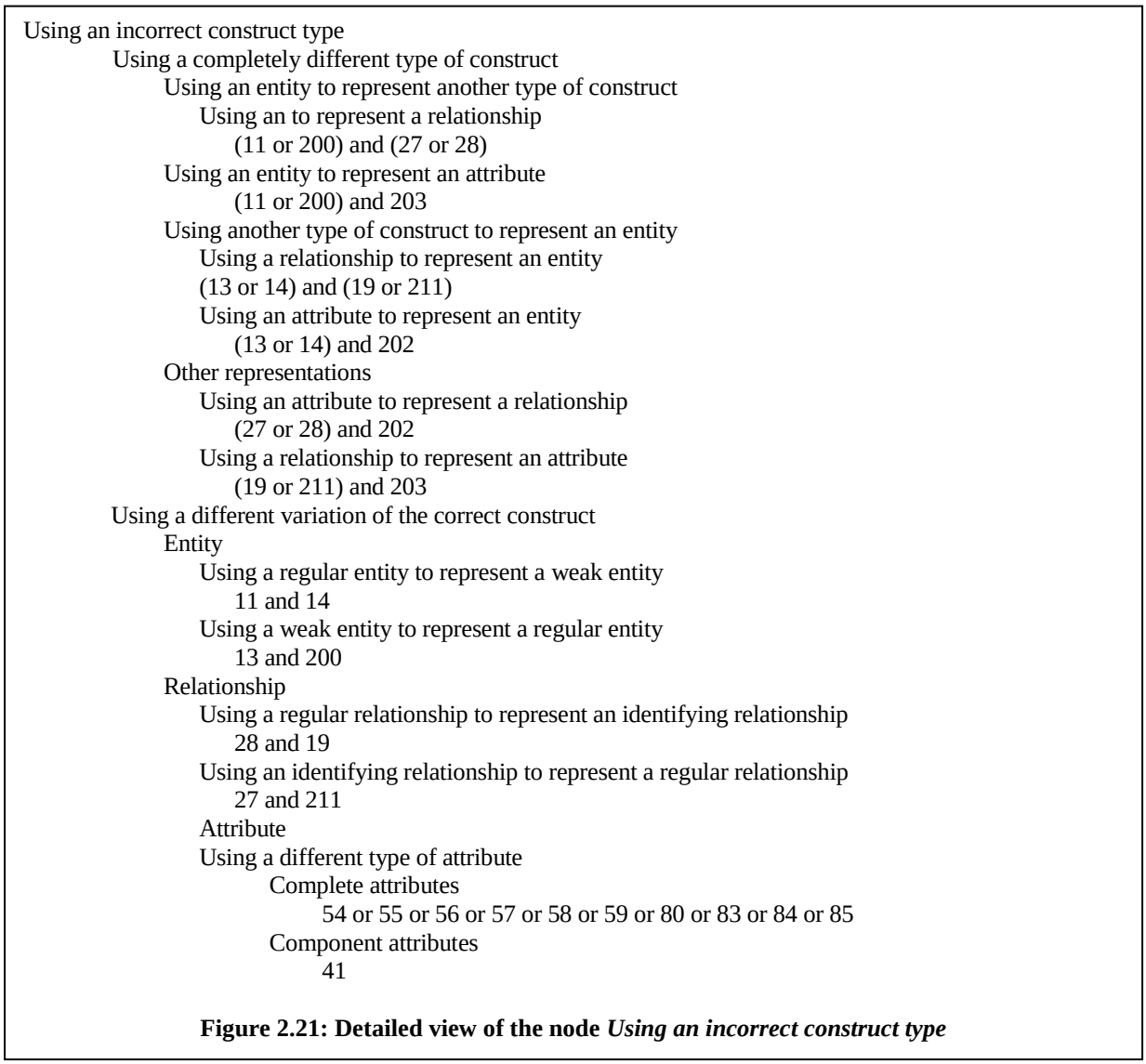


Figure 2.21: Detailed view of the node *Using an incorrect construct type*

Including constraints in the hierarchy: Some leaves of the error hierarchy contain a single constraint for a single node. For example constraint 37 that focuses on binary relationships (it

checks whether there is a matching binary relationship in the student solution for every binary relationship in the ideal solution) is categorised under the node *Missing constructs* (Figure 2.22). The more complex case is having multiple constraints for a single node. This can happen due to different reasons: (i) Multiple constraints are needed to define an error precisely; (ii) multiple constraints specify multiple instances of an error type; and (iii) A single error type is covered in multiple constraints.

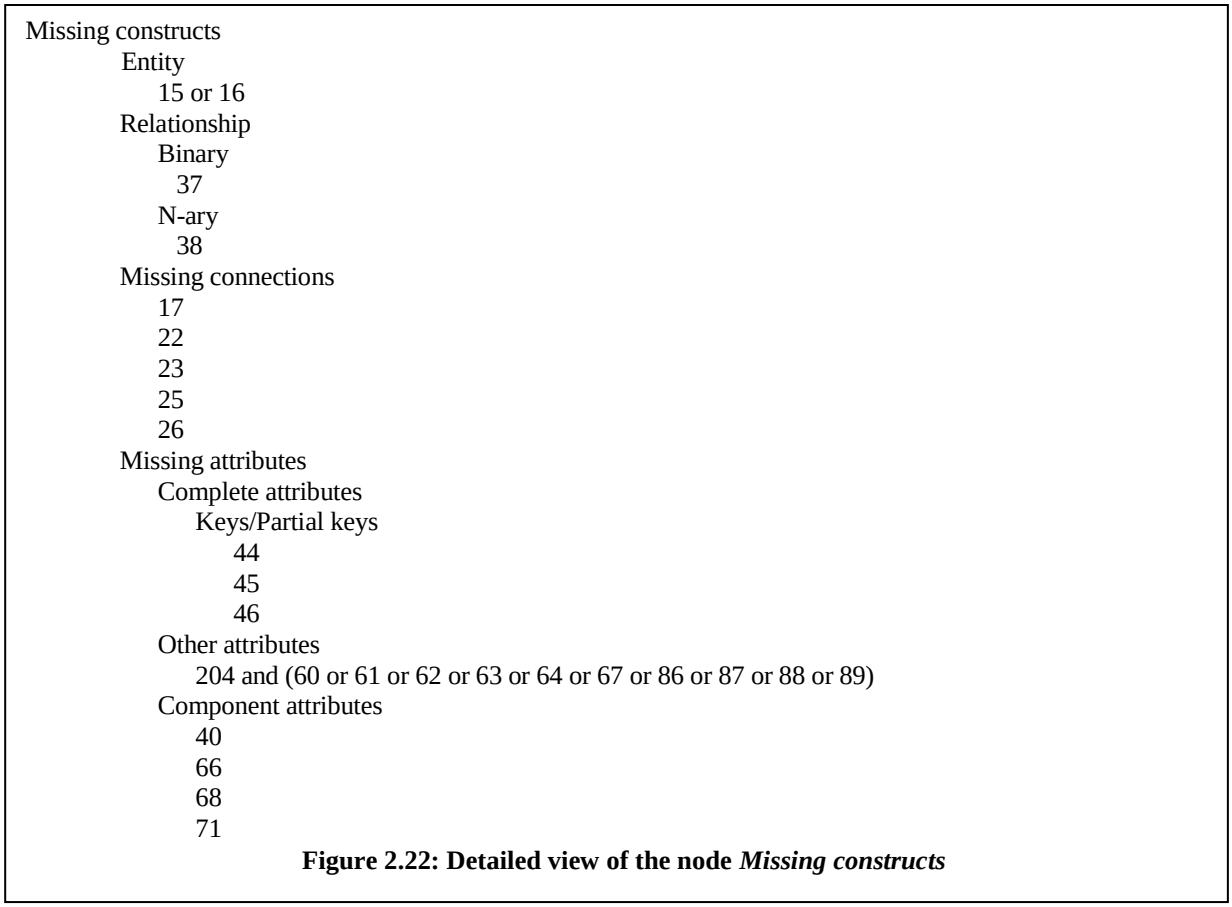


Figure 2.22: Detailed view of the node *Missing constructs*

To illustrate case (i), consider the regular entity CHAPTER in the student solution in Figure 2.7(a). Modelling CHAPTER as a regular entity violates constraints 11 and 14. Constraint 11 deals with extra regular entities in the student solution which should be modelled using some other type of construct. (i.e. constraint 11 checks for the existence of a matching regular entity in the ideal solution for every regular entity in the student solution). Constraint 14 focuses on weak entities which are represented as some other type of construct in the student solution (i.e. constraint 14

checks for the existence of a matching weak entity in the student solution for every weak entity in the ideal solution). This situation is described by the node *Using a regular entity to represent a weak entity* which is a child node of *Using a different variation of the correct construct* (Figure 2.21). The two constraints (i.e. 11 and 14) are combined with 'and' to specify that both constraints need to violated by the same construct i.e. CHAPTER is an extra regular entity in the student solution (constraint 11) and should be modelled as a weak entity (constraint 14).

As another example, consider the error using an attribute to represent an entity. A student can make this error in two possible ways: (i) using an attribute to represent a regular entity (ii) using an attribute to represent a weak entity. When a regular entity is represented as an attribute, constraints 13 (focuses on regular entities which are represented as some other type of construct in the student solution) and 202 (focuses on attributes in the student solution that are used to represent another type of construct) will be violated. Similarly when a weak entity is modelled as an attribute, constraints 14 (deals with weak entities which are represented as some other type of construct in the student solution) and 202 will be violated. As a result, all these constraints are grouped together as *(13 or 14) and 202* and assigned to the node *Using an attribute to represent an entity* (Figure 2.21).

In case (ii), multiple constraints that specify multiple instances of the same error type can be grouped together so that a single dialogue customised to reflect the current error in the student solution can be used. For example, constraints 54, 55, 56, 57, 58, 59, 80, 83, 84 and 85 are grouped together to specify the error *Using a different type of attribute* (Figure 2.21). Constraint 54 deals with simple attributes connected to an entity, modelled as some other type of attribute (i.e. constraint 54 checks for the existence of a matching simple attribute in the student solution for every simple attribute in the ideal solution). Similarly constraint 80 deals with simple attributes belonging to a relationship. Constraint 57 deals with derived attributes connected to an entity whereas constraint 87 deals with derived attributes belonging to a relationship. As all these constraints denote different instances of the same error type *using a different type of attribute* they are grouped together using 'or'.

The final scenario involves an error type covered in multiple constraints. Incorrect cardinality and participation are two such errors. Two cardinalities have to be specified for each binary relationship. For example two cardinalities for the binary relationship CONTAINS are: (i) the

cardinality between TEXT_BOOK and CONTAINS (ii) the cardinality between CHAPTER and CONTAINS. In addition to having to specify two possible cardinalities for each binary relationship, there are two possible values (1 or N) for each cardinality. Therefore, constraints 92 and 93 deal with cardinality being '1', whereas constraints 94 and 95 deal with the value N. As a result, all these constraints are grouped together using 'or' and assigned to the node *Cardinality* which is a child node for *Errors dealing with cardinalities and participation* (Figure 2.23). Similarly there are four constraints that deal with the two possible values for participation (i.e. total and partial) Constraints 96 and 97 focus on total participation whereas constraints 98 and 99 deal with partial participation. All these constraints are grouped together (Figure 2.23).

Errors dealing with cardinalities and participation
- Cardinality
  - Regular relationship
    - 92 or 93 or 94 or 95
  - Identifying relationship
    - 33
  - Recursive relationship
    - 102 or 103 or 104
- Participation
  - Regular relationship
    - 96 or 97 or 98 or 99
  - Identifying relationship
    - 34
  - Recursive relationship
    - 105 or106 or 107

Figure 2.23: Detailed view of the refined node *Errors dealing with cardinalities and participation*

Tutorial dialogues: Error remediation is facilitated through tutorial dialogues. A dialogue is designed for each error type (i.e., each leaf node in the hierarchy). As the domain model of constraint-based tutors is represented as a set of constraints, violations of constraints indicate the domain concept that the student has difficulty with. Each dialogue discusses the domain concept associated with an error as well as providing an opportunity to reflect on that error within the current problem context.

There are two types of dialogues: (i) Single-level dialogues and (ii) Multi-level dialogues. Single-level dialogues handle errors associated with simple syntax errors for which a detailed feedback message is sufficient to explain the error. Hence these dialogues are limited to a single

feedback message. There are 12 such dialogues. An example of a single level dialogue is "You have connected an entity A to entity B directly. Entities cannot be directly connected to each other". This is associated with constraint 7 (Suraweera, 2001). Multi-level dialogues handle other more complex errors for which a series of prompts is necessary to guide students to self-explain both domain concepts and problem-solving steps. These dialogues consist of four levels:

(i) Informs the student about the incorrect modelling decision and asks to focus on the corresponding domain concept

(ii) Prompts the student to understand why his/her modelling decision is incorrect

(iii) Prompts the student to specify the correct modelling decision

(iv) Prompts the student to review the domain concept learnt through a review question

There are two exceptions to the 4-level dialogues. The first refers to those dialogues that focus on the concepts of cardinality and participation; they consist of only three levels. The second exception is the different focus of the level 2 prompt when discussing missing constructs. Both these exceptions will be discussed in detail at the end of this section.

Dialogue presented in Figure 2.24 discusses the error that CHAPTER has been modelled as a regular entity. This error was one of many errors identified in the evaluation of the student solution in Figure 2.7(a). This error was selected as the simplest error to discuss as it is the first one in the error hierarchy that corresponds to one of the violated constraints.

We now discuss what happens in each stage of the dialogue. In the first stage of the dialogue, the system informs the student about the error and focus on the corresponding domain concept (KERMIT-SE1 in Figure 2.24). If the student requests the correct explanation or fails to provide the correct one, the student is given more help by explaining the domain concept or by guiding him/her through simpler questions (KERMIT-SE2 in Figure 2.24). If he/she still fails to understand the mistake through the simple prompt provided by the tutor, the correct answer is provided as a last resort. Providing the correct explanation at any stage, moves the dialogue to the next stage and the next prompt is displayed.

In the second stage of the dialogue, the student is guided to explain why his/her modelling decision is incorrect (KERMIT-SE4 in Figure 2.24). If it is difficult for the student to understand why his/her decision is incorrect (Student3 in Figure 2.24), the tutor guides the student to search for more information in the problem statement. For instance, when the student fails to understand

why modelling CHAPTER as a regular entity is incorrect, he/she is asked to search for any key attributes for the entity type CHAPTER (KERMIT-SE5 in Figure 2.24). As students need to be actively engaged in finding out whether there are any key attributes for CHAPTER, the possible answers are not given for the question immediately. Instead, the list of possible answers is displayed only when the student informs the tutor that he/she is ready to continue with the self-explanation process. By delaying the display of possible answers when students are expected to search for information in the problem statement, they are encouraged to actively engage in understanding why their modelling decision is incorrect. Then he/she is given the opportunity to understand that CHAPTER cannot be a regular entity because it does not have a key attribute (Student4 in Figure 2.24).

In the third stage, the student is asked to specify the correct modelling decision (KERMIT-SE6 in Figure 2.24). Again, if he/she fails to specify the correct the modelling decision, the tutor provides the correct answer with an explanation.

In the final stage of the dialogue, the student is given a review question providing another

KERMIT-SE1: You've modelled *CHAPTER* as a regular entity. Can you tell me the general rule to decide whether something is a regular entity?
Student1: *I don't know*

KERMIT-SE2: Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity?
Student2: *partial key attribute*

KERMIT-SE3: This is a fundamental concept you need to remember. A key attribute is used to uniquely identify instances of a regular entity.
KERMIT-SE4: Now tell me why modelling *CHAPTER* as a regular entity is incorrect?
Student3: *I'm not sure*

KERMIT-SE5: Let me help you a bit. Check carefully whether any of the attributes of *CHAPTER* can uniquely identify the instances of *CHAPTER? Please press Continue if you want more clarifications.*
{After Continue button is pressed}
Can you tell me the attribute that you've identified as a key attribute for *CHAPTER?*
Student4: *There is no attribute that can be a key for CHAPTER.*

KERMIT-SE6: Great! Now tell me the best way to model *CHAPTER*?
Student5: *A weak entity*

KERMIT-SE7: Well Done! Now select the correct pair of words to fill in the blanks so that you'll remember what you learnt.
A…….. is used to uniquely identify instances of a …….. entity.
Student-6*:* *key attribute, regular*

KERMIT-SE8: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Figure 2.24: Hypothetical dialogue in KERMIT-SE

opportunity to understand the concept discussed (KERMIT-SE7 in Figure 2.24). Various types of review questions are used to maintain the students' motivation. The review questions can be simple questions, fill-in-the-blanks, or true-false questions. If the student has difficulty in answering the review question correctly, the correct answer is provided.

Each dialogue is presented in a tree form, in which each distinct node has a prompt and a list of possible explanations. The tutorial dialogue starts from the root node and finishes at one of the leaf nodes. The specific path taken depends on the accuracy of the explanation selected by a student for each question. An explanation selected can be correct, incorrect or a request for the correct explanation (by selecting an option such as "I'm not sure" or "I don't know"). A correct explanation triggers the selection of the prompt associated with the left child of the current node. The other two cases (i.e. incorrect explanation or a request for the correct one) trigger the selection of the prompt associated with the right child of the current node. The response of the dialogues is the same for an incorrect explanation and for a request for a correct explanation to keep the implementation simpler. For example, the next prompt after the prompt "Now tell me why modelling CHAPTER as a regular entity was incorrect?" (KERMIT-SE4 in Figure 2.24) is "Let me help you a bit. Check carefully whether any of the attributes of CHAPTER can uniquely identify the instances of CHAPTER (KERMIT-SE5 in Figure 2.24) for both cases. The limitation of this implementation is that the system does not explicitly indicate that the explanation is incorrect. Instead the generic phrase "Let me help you a bit" is used in both situations.

There is evidence that immediate feedback on students' responses has the potential to enhance learning in an ITS (Aleven, Popescu, & Koedinger, 2002). Therefore, it is important to provide feedback on the accuracy of the self-explanations provided by students during problem-solving. In addition to providing feedback on students' responses, phrases such as "Well done", "Great" and "Good job" (KERMIT-SE7 in Figure 2.24) are used in the dialogues to encourage students to actively engage in dialogues. Furthermore, when a student fails to provide the correct self-explanation he/she is encouraged by using phrases such as "Let me help you a bit" (KERMIT-SE4 in Figure 2.24), instead of phrases like "Your answer is incorrect. Please try again". Even though the dialogues contain a series of questions, students can correct a mistake as soon as they realise it without going through the entire dialogue. Therefore, knowledge construction is facilitated for

students who have misconceptions or gaps in their knowledge without disrupting others (Bunt et al., 2004).

Dialogues with three levels: As mentioned earlier, dialogues that discuss cardinality and participation consist of only three levels. This is because only two possible values exist for cardinality (i.e. 1 or N) and they are the only ones allowed by the interface (Figure 2.2). As soon as the student is made aware that the cardinality between an entity and relationship is incorrect, the correct answer becomes clear. Asking students to specify the cardinality using another prompt might potentially demotivate them if they want to resume problem-solving straightway. When they correctly explain why the specified cardinality is incorrect, the student is given the final prompt of the dialogue. For example, the response will be "Great Job! I guess you know how to correct the mistake now. Before starting to make changes, try to answer this question. What is the correct question to ask when deciding the participation between entities E1 and E2 in a binary relationship?" Similarly participation also has two possible values: total and partial. As soon as the ITS tells the student that the specified participation is incorrect, then the correct answer becomes apparent. Thus the dialogues that discuss errors related to participation also have only three levels.

Different Level-2 prompt when discussing missing constructs: The level-2 prompt focuses on why a student modelling decision is incorrect. However this is not applicable when the error is about missing constructs. In such situations, the student is asked to specify the type of construct that is missing.

Evaluation: An evaluation study was conducted at the University of Canterbury, to investigate whether guided self-explanation would facilitate acquisition of both procedural and conceptual knowledge in the domain of conceptual database design. The experiment involved 125 second-year university students enrolled in an introductory database course. The experimental group used KERMIT-SE, while the control group used a modified version of KERMIT with limited feedback. Both groups received a list of all errors for their solutions, and could request the ideal solution. While the experimental group received a dialogue discussing a selected error, their peers in the control group received a detailed feedback message on a chosen error. This detailed

feedback message corresponds to the Detailed Hint in original KERMIT (Section 2.2.3). None of the other feedback levels were available to the control group.

The pre- and post-tests consisted of two questions each, of equal difficulty. The first question required an ER model to be designed for the given requirements, whereas the second question asked for an explanation on the design decisions for the given ER model. The first question was used to measure their problem-solving capabilities and the second question their self-explanation abilities.

The results showed that performance of both experimental and control groups improved. Furthermore, a significant improvement was achieved only by the control group. However, their pre-knowledge (pre-test score) was lower than that of the experimental group so they had more room for improvement. Analysis of the participants' logs indicated that only 35% of the students in the experimental group had gone through at least one dialogue as some participants used only the list of all errors to correct their solutions. We compared the performance of these students (*self-explainers*) with the other students who have not self-explained (*non self-explainers*). Although both groups improved after interacting with KERMIT-SE, the improvement was not statistically significant.

Analysis of pre-and post-test performance on each question for self-explainers and non self-explainers revealed that the improvement in procedural knowledge (based on question 1 in the pre and the post-tests) of non self-explainers is better than their peers although not statistically significant. This suggests that guiding students towards the ideal solution through general feedback messages provided by the cut-down version of KERMIT help them to improve their procedural knowledge. However, the better performance of self-explainers on the second question of the post-test that focused on conceptual knowledge, was statistically significant. It is interesting to note that the performance of the non self-explainers decreased after using the cut down version of KERMIT which did not provide any self-explanation support. Therefore, these results suggest that the self-explanation support provided by KERMIT-SE has been a significant contributor towards improving the conceptual knowledge of the students.

We wanted to investigate whether the system is more beneficial to less able students. We divided the self-explainers and the non self-explainers into *more* and *less* able groups based on their performance in the pre-test. The *more able* group comprised of those students who scored above the

mean score for all participants and the remaining students formed the less able group. The performance of less able students in both groups improved significantly whereas there was a decrease in the performance of more able students.

2.5 Discussion

Table 2.1 summarises the systems using the two dimensions discussed in Section 2.3 (main activity supported by the system and the form of student responses). The focus of our research is on systems that use problem-solving as the main activity and use dialogues as the additional support (systems included in first column of Table 2.1). Such systems allow learners to either respond in free form natural language or using a pre-specified set of options. The type of tasks supported by these problem-solving environments range from well-defined tasks including physics and mathematics, to ill-defined tasks such as conceptual database design. All the systems discussed here except KERMIT-SE provide learning support for well-defined tasks.

Table 2.1 lists three significant systems that engage students in discussions as the main learning activity (top right cell of Table 2.1). The original version of AutoTutor was developed to teach computer literacy and it has evolved to cover multiple domains (Graesser, 2011). The other two systems support learning qualitative physics. As learning opportunities provided by dialogues can be severely limited if the students respond via a pre-specified list of options, there have been no research attempts develop such systems.

Table 2.1: Classification of Dialogue-based systems

		Main activity	
		Problem-solving	Discussion
Form of response	Free form responses in natural language	CIRSCIM-Tutor (Evans & Michael, 2006) Atlas-Andes (VanLehn et al., 2010) Geometry Explanation Tutor (Aleven, Popescu, et al., 2003) Ms. Lindquist (Heffernan et al., 2008)	AutoTutor (Graesser, 2011) Why2-Atlas (Jordan et al., 2006) Why2-AutoTutor (VanLehn et al., 2007)
	Selecting from a pre-specified set of options	NORMIT-SE (Mitrovic, 2005) Geometry Explanation Tutor (Aleven & Koedinger, 2002) KERMIT-SE (Weerasinghe & Mitrovic, 2006).	

These systems also differ in how they customise the selection of each dialogue. Existing systems use pre-specified strategies based on the history of the tutoring session of a student. For example, Geometry Explanation Tutor focuses on geometry aspects in the student explanation that is not relevant to the current step before missing ones. Why2-Atlas prioritizes tutorial goals in the following order: misconceptions, self-contradictions, errors and incorrect assumptions and missing mandatory points.

There are two major limitations in the existing systems. Firstly, none of the dialogue-based systems that use problem-solving as the main activity and dialogues as additional support have been used in multiple domains. The other limitation is that selecting the dialogue is based on simple heuristics focusing on the performance of the student, not on the history of a student's evolving knowledge. In this research we propose a model that provides adaptive dialogue support in multiple domains. Our model uses a novel approach that combines the history of the tutoring session with the student model. This model is developed to support problem-solving with dialogues and expect student responses via a pre-specified list. We present the model in Chapter 3.

Chapter 3

A General Model Supporting Tutorial Dialogues

There have been several research attempts to investigate the effectiveness of tutorial dialogues in human one-on-one tutoring and then to replicate the same effectiveness within ITSs. However all these attempts have been implemented in a single domain and none of them focused on developing a model to facilitate dialogue support across domains. Hence the two main research questions are: (a) Is it possible to provide adaptive dialogue support across domains? (b) Are adaptive dialogues better than non-adaptive dialogues in learning both domain knowledge and procedural knowledge?
In this chapter, we discuss how we extend the model developed for my M.Sc. thesis to provide adaptive dialogue support in both ill-defined and well-defined tasks. In Section 3.1, we discuss two dimensions for classifying instructional domains and tasks, which provides the foundation for this chapter. We then discuss how we plan to evaluate the main contribution of this research followed by details of our model in Section 3.3. The goal of this model is to provide adaptive dialogue support across instructional domains and tasks. This model emulates the behaviour of human tutors by providing adaptive dialogues during problem solving in response to errors. This model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level, an error type is associated with one or more violated constraints, which forms leaves of the hierarchy. The error types are then grouped into higher level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the pedagogically most suitable error for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to customise the dialogue for a student at a particular time in the tutoring session. Adaptation rules also individualize the dialogues to suit the student's knowledge and reasoning skills. In response to the generated dialogue, the learner is able to provide an explanation by selecting the correct option from a list.

3.1 Difference between a domain and a task

We first need to differentiate between instructional tasks and domains, to understand the different types of instructional tasks. We use the term *domain* to refer to declarative domain knowledge, or the domain theory, while an *instructional task* is the task the student is learning, in terms of problem-solving skills. In order to learn a particular instructional domain, the student needs to learn the relevant declarative knowledge (i.e. the domain theory), and in many domains also needs to acquire problem-solving skills. It is important to make a clear distinction between these two types of learning, declarative knowledge versus problem-solving skills for the discussion of ill-definedness/well-definedness. ITSs are almost exclusively problem-solving environments, based on the assumption that students have learnt the declarative knowledge from direct instruction (lectures, books and/or peers) and need opportunities to practice their problem-solving skills (Koedinger et al. 1997; Mitrovic et al. 2007). Most ITSs provide lots of problem-solving opportunities and only occasionally give direct instruction, in the form of examples or definitions of the concepts used as in (Aleven, Koedinger, & Cross, 1999). There are also ITSs that provide instructional material in addition to problem-solving support, such as ELM-ART (Brusilovsky & Weber, 2001), but we will focus on problem-solving as the main instructional activity here.

Most researchers equate ill-definedness of a task with that of the underlying domain theory, and provide examples of ill-defined domains, such as essay writing (Aleven, Ashley, Lynch, & Pinkwart, 2006; Aleven, Ashley, Lynch, & Pinkwart, 2007; Aleven, Ashley, Lynch, & Pinkwart, 2008). Commonly used examples of well-defined domains are mathematics and physics. However, there seems to be no differentiation between the characteristics of domains versus tasks.

3.1.1 Classifying domains

We proposed that two orthogonal dimensions need to be considered when discussing ill-definedness: the domain, and the task (Mitrovic & Weerasinghe, 2009). Domains vary in terms of their underlying domain theories. There are many domains covered by ITSs that are completely well-defined, such as many areas of mathematics, physics and chemistry. Instructional tasks that

these ITSs teach are also well-defined: for example, adding fractions, multiplying mixed fractions, solving equations for unknowns, or balancing chemical equations. The student is taught the theory, as well as the procedure (i.e. the algorithm) to use to solve problems. Such domains are in the WDWT (well-defined domain, well-defined task) quadrant in Figure 3.1.

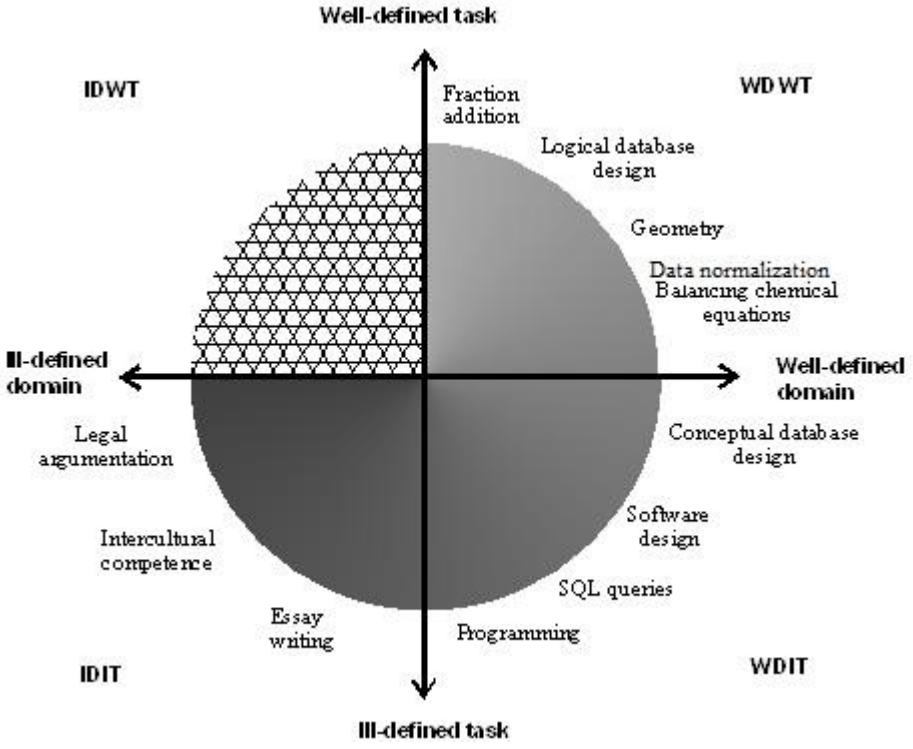


Figure 3.1: The space of instructional domains and tasks, from (Mitrovic & Weerasinghe, 2009)

However, if the domain is well-defined, that does not necessarily mean that instructional tasks in that domain will also be well-defined. As an illustration, let us focus on the domain of database design (Elmasri & Navathe, 2010). Conceptual database design is a task of converting the database requirements into a high-level description of the database, most often expressed in terms of the Entity-Relationship (ER) data model (Elmasri & Navathe, 2010). On the other hand, logical database design is a process of converting an ER diagram into a relational schema, thus requiring an understanding of the relational data model. Both the ER and relational data models are well-defined: they consist of a small number of components with well-defined syntax and semantics. Although the ER model itself is well-defined, the task of developing an ER schema for a particular

database (i.e. conceptual database design) itself is ill-defined: the initial state (i.e. the set of requirements) is usually underspecified and ambiguous, there is no algorithm to use to come up with the solution, and finally the goal state is also underspecified, as there is no simple way of evaluating the solution for correctness. Therefore, conceptual database design belongs to the WDIT (well-defined domain, ill-defined task) quadrant in Figure 3.1. Logical database design (also known as database mapping), however, is well-defined, as there is a simple deterministic algorithm which guarantees good solutions (shown in the WDWT quadrant in Figure 3.1). Other examples for the WDIT (well-defined domain, ill-defined task) quadrant include programming and writing SQL queries: although the relevant languages are well-defined, the task of converting the problem statement into a program is ill-defined.

Many domains are ill-defined, such as essay writing. In that case, the declarative knowledge is incomplete: it specifies how to structure the essay, how to present arguments, and also defines writing styles. The domain theory in this case is ill-defined, as is the task itself (writing the essay), as illustrated in the IDIT (ill-defined domain, ill-defined task) quadrant in Figure 3.1. Psychological diagnosis is an example of a well-defined task associated with an ill-defined domain theory; a task that can be placed in the IDWT (ill-defined domain, well-defined task) quadrant. The task is well-defined due to the procedures defined by the experts. However the associated domain theory is ill-defined as human understanding of the domain is still being formed. The two dimensions are continuous; there is a spectrum arranging domains from ill- to well-defined ones, as well as another spectrum for instructional tasks. There are some dependencies between them, as ill-defined domains usually involve ill-defined tasks, but the contrary is not necessarily so. (i.e. ill-defined tasks do not necessarily involve ill-defined domains.)

3.1.2 Classifying instructional tasks

When discussing the definedness of decision-making tasks, there are four important factors to consider (Reitman 1964; Taylor 1974): start state, goal state, and the transformations (i.e. the problem-solving procedure), as well as the decision maker's familiarity with each of the factors. Decision making is similar to problem-solving. i.e. decision making starts with the current state, involve some transformations of the current state to reach the desired goal state. Due to this

similarity, we adopt these factors in our classification of instructional tasks. In addition, we add another one: the existence of a correct solution.

The initial state is presented to the student in the form of a problem statement. Instructional tasks taught to younger students most often have well-specified problem statements – e.g. simple arithmetic tasks, equation solving and other tasks in science.

Problem statements for more challenging tasks can be less specified: in a typical university-level mechanics problem, the text of the problem does not specify all the forces acting on a given body. In conceptual database design or software design, the student is given a set of requirements, which is often incomplete and/or ambiguous. To deal with such problems, the student needs to use not only declarative domain knowledge they learnt previously, but also his/her world knowledge in order to eliminate ambiguities and (when necessary) add missing information. Therefore, in order to deal with ill-defined problem statements, the student has to process the given information in order to complete the specification (and therefore turn the problem statement into a well-defined one).

Goal states can also be well- or ill-defined. In easy tasks, the student is clear about the form of the final solution. For example, the student had learnt that there were two solutions for a quadratic equation before attempting to solve any equations. When adding two fractions, the student knows that the solution should be (in the general case) another fraction. Additionally, the student can easily check whether the solution is correct or not with the help of a teacher or a calculator. However, in design tasks, there is little information about the goal state.

The goal state in such tasks is defined in a very abstract way; for example, in database design the goal state is defined as an ER diagram that is syntactically correct and matches the given requirements. Therefore, there is no simple test to use to check for correctness; the student can only apply the declarative knowledge he/she possesses in order to evaluate the solution produced. Another important issue is whether there is a stopping criterion – how can the student tell whether he/she is finished solving the problem? A well-defined goal possesses a stopping criterion which is easy to apply, while the ill-defined ones do not, and the student is again left to apply the constraints from the declarative knowledge in order to evaluate the solution.

Transformations or the problem-solving procedure is another important factor. Some instructional tasks have a well-defined algorithm to apply to the initial state to derive the goal state.

We have previously mentioned several tasks from mathematics, physics or chemistry with well-defined algorithms. In such situations, the student task is (relatively) easy: the student needs to memorize the algorithm and apply it correctly.

Other similar examples involve some engineering tasks involving calculations. However, there is a very important subclass of tasks that deal with design. Design is in general ill-defined, as there are no algorithms to use to transform the initial state into the goal state. In addition, the start state is underspecified, and the goal state is defined in terms of highly abstract features. Design tasks typically involve huge domain expertise, and large, highly structured solutions (Goel & Pirolli, 1993). Typical examples of design tasks include architecture, software design, mechanical engineering and music composition. Two possible methods to reach a solution are to apply a heuristic procedure or to use analogical reasoning. The first method, heuristic rules can guide the student, but in general the student needs to apply the constraints from the domain theory. The second method, analogical reasoning involves comparing the current problem to those previously solved to deal with the complexity.

Finally, some researchers believe that ill-defined tasks are those that have no correct solution, but rather a family of solutions which can all be deemed correct. This is true of the extreme cases, such as essay writing: there might be any number of very good essays on a specified topic. In design tasks, there are often several (or even many) solutions that are all equally good. However, in a teaching situation, the teacher often has a good pedagogical reason for preferring one particular solution over the others. For example, in SQL there are often several correct queries for the same problem, differing from each other in the constructs used (please note that there is a lot of redundancy in SQL and, therefore, multiple ways of satisfying the same requirements). Even in such a task, the teacher may prefer one of those solutions among others; for example, the teacher may want to illustrate the use of a particular predicate. Therefore, it is still possible to nominate one "ideal" solution (chosen for pedagogical reasons) without compromising the quality of the whole ITS, as long as the ITS is capable of identifying other alternative solutions students may come up with as correct.

Table 3.1 presents factors of instructional tasks and presents a few examples, categorized with respect to the factors discussed.

Table 3.1. Some examples of instructional tasks and their domains, adapted from (Mitrovic & Weerasinghe, 2009)

Instructional task	Domain	Problem specification (initial state)	Goal specification (goal state)	Problem-solving procedure	Correct solution
Fraction addition	Well-defined	Well-defined	Well-defined	Well-defined	Only one
Balancing chemical equations	Well-defined	Well-defined	Well-defined	Well-defined	Only one
SQL queries	Well-defined	Ambiguous/ incomplete	Abstract	None	Multiple
Software design	Well-defined	Ambiguous/ incomplete	Abstract	None	Multiple
Essay writing	Ill-defined	Abstract	Abstract	None	Multiple
Legal argumentation	Ill-defined	Abstract	Abstract	None	Multiple
Intercultural competence	Ill-defined	Abstract	Abstract	None	Multiple
Conceptual database design	Well-defined	Ambiguous/ incomplete	Abstract	None	Multiple
Logical database design (database mapping)	Well-defined	Well-defined	Well-defined	Well-defined	Only one
Data normalization	Well-defined	Well-defined	Well-defined	Well-defined	Multiple

3.1.3 Tasks selected for the research

Several tasks ranging from ill-defined to well-defined were chosen to explore the applicability of the model supporting dialogues. We tested our model in conceptual database design, logical database design, data normalization and fraction addition. Conceptual database design was discussed in Section 2.2.3. We now discuss logical database design, data normalization and fraction addition in detail.

Logical database design: A database schema is the end result of the conceptual database design process. It is detailed enough to be used by database developers as a blueprint for implementing the database. The information contained in the database schema will be used to define the relations, primary and foreign keys. As there are no database management systems (DBMS) based on conceptual data models, the high-level database schema needs to be translated to a schema supported by the chosen DBMS; this process is known as the logical database design (Elmasri &

Navathe, 2010). Logical database design is usually taught in introductory database courses at the undergraduate level, via a series of lectures and paper-based exercises.

Logical database design is a well-defined task: a student is expected to follow the algorithm consisting of seven well-defined steps. Each step in the algorithm maps one concept from the ER diagram by either creating a new relation, or altering previously created relations by adding foreign keys and attributes. The seven steps of the algorithm are as follows:

- Step 1: Map all regular entity types and their simple attributes.
- Step 2: Map all weak entity types and their simple attributes.
- Step 3: Map all 1:1 relationship types.
- Step 4: Map all 1:N relationship types.
- Step 5: Map all M:N relationship types.
- Step 6: Map all multivalued attributes.
- Step 7: Map all N-ary relationship types.

Even though the algorithm is well-defined and short, students typically find it hard to learn and apply consistently. Therefore, providing adaptive tutorial dialogue support to discuss the errors in student solutions would be beneficial to acquire a deep understanding of this domain.

Data Normalization: Data normalization is the process of refining an existing relational database schema to ensure that all relations are of high quality (Elmasri & Navathe, 2010). Normalization is normally taught in introductory database courses at undergraduate level. Relevant concepts are introduced in a series of lectures followed by paper-based exercises. Data normalization is very theoretical compared to other topics in the area of databases due to the many algorithms and multiple definitions that need to be learnt. As a result, students find it very difficult to gain a robust understanding of normalization (Kung & Tung, 2006; Phillip, 2007). Students are required to have a good understanding of the relational data model, especially various types of keys (primary, candidate, foreign keys and superkeys). They are also expected to understand the concept of functional dependencies, which specify constraints on the values of sets of attributes. The

definitions of various normal forms are also important and finally the various algorithms that are required for normalizing relations must be understood (Elmasri & Navathe, 2010).

Data normalization is a well-defined instructional task. Each problem consists of a relation whose schema is provided, and a set of functional dependencies (which does not have to be complete). For example, the student might be given a relation R(A, B, C, D, E) (please note that typically students are not given semantics of the attributes, although that can also be provided) and the following set of functional dependencies:

{A→BC, CD→E, AC→E, B→D, E→AB}

The normalization procedure as implemented in NORMIT (Mitrovic et al., 2012), an ITS that teaches data normalization consists of eleven tasks described below. Please note that we refer to elements of the procedure as *tasks* rather than *steps*, as each of them contains a number of actions the student has to perform, including in some cases relatively complex algorithms. Therefore we refer to them as tasks to make it clear that the tasks are relatively complex compared to what is generally assumed by a step in the ITS research.

The first eight tasks are necessary to determine the highest normal form the relation is in. If the relation is not in BCNF, the student needs to apply the relational synthesis algorithm to derive an improved database schema. The relational synthesis algorithm is implemented via tasks 9-11.

1. Identify the candidate keys for the given table. There may be one or more keys depending on the given relations. For example, A, E, BC and CD are the candidate keys for the above problem. To explore various possibilities for candidate keys, the student can compute the closure of a set of attributes. Therefore, finding the closure is a subtask of task 1.
2. Find the closure of a given set of attributes. This task consists of an algorithm that is used in several other tasks, including finding candidate keys. In the above problem, to specify BC as the key for relation R, we need to determine that its closure consists of all attributes of relation R.
3. Identify prime attributes. Prime attributes are those attributes that belong to any candidate keys. In the above problem, A, B, C, D and E are the prime attributes.
4. Simplify functional dependencies (FDs) using the decomposition rule, if necessary. In this task, a functional dependency with more than one attribute on the right-hand side (RHS) is

replaced with as many FDs as there are attributes on RHS. Each new FD has the same left-hand side as the starting one, and just a single attribute as its RHS. In the above problem, A → BC would be replaced by the two FDs: A → B and A → C. Similarly E → AB would be replaced by E → A and E → B.

5. Determine the normal forms for the given relation. In this task, the student needs to determine whether the relation is in 1NF, 2NF, 3NF or BCNF. In this step, the expected answer is limited to Yes or No responses.
6. If the student indicated that the relation is not in 2NF, he/she expected to identify FDs that violate that form (i.e. partial FDs).
7. If the student indicated that the relation is not in 3NF, he/she needs to identify FDs that violate that form (i.e. transitive FDs).
8. If the student specified that the relation is not in BCNF, he/she will be asked to identify FDs that violate that form (i.e. FDs which do not contain a superkey on their left hand sides (LHSs).
9. For relations that are not in BCNF, reduce LHS of FDs. This task checks whether some of the attributes on the LHS can be dropped while still having a valid functional dependency.
10. Find a minimal cover, which is a minimal set of FDs that still capture all the necessary information. In this task, the students need to apply the algorithm for checking whether a FD is redundant (and therefore can be dropped from the minimal cover) or not.
11. Decompose the table using the minimal cover.

Fraction addition: Adding two fractions is an important part of the mathematics curriculum. In order to calculate the sum of two fractions correctly, a series of steps have to be followed sequentially. There are different approaches for fraction addition, but we used an approach consisting of four steps. Initially, the least common denominator (LCD) of the two fractions has to be found. Then the two fractions have to be converted to have LCD as their denominator. The sum of the numerators of the converted fractions becomes the numerator of the resulting fraction, with LCD as its denominator. Finally, the resulting fraction has to be simplified, if possible to produce the final result.

3.2 Evaluation of main research contributions

As the evaluation of interactive adaptive systems (IASs) has been acknowledged to be a complicated endeavor, many researchers emphasise a layered approach (Brusilovsky, Karagiannidis, & Sampson, 2001; Paramythis, Totter, & Stephanidis, 2001; Weibelzahl, 2001), i.e. evaluating adaptivity by decomposing and evaluating it in a piece-wise manner. One of the recent approaches suggested by Paramythis and colleagues (2010) unifies the common themes of the previous approaches and is known as the *layered evaluation of interactive adaptive systems*. Table 3.2 presents the main layers of adaptation identified by this framework together with the suitable evaluation methods. Even though it is argued that each adaptation layer needs to be evaluated explicitly, all layers cannot be "isolated" and evaluated separately in all systems. The relevance of each layer depends on the nature of the IAS.

Table 3.2: Adaptation layers and evaluation methods for each layer (adapted from Alexandros Paramythis, Weibelzahl, & Masthoff, 2010)

Adaptation layer	Description	Evaluation methods
1. Collection of input data (CID)	Gathering user interaction data along with any other data (available through non-interactive sensors) related to the interaction context	Data mining, Play with layer, Simulated users, Cross-validation
2. Interpretation of the collected data (ID)	Specifying meaning within the system for the raw input data previously collected	Data mining, Heuristic evaluation, Play with layer, Simulated users, Cross validation
3. Modelling of the current state of the "world" (MW)	Deriving new knowledge about the user, the interaction context, etc. Subsequently this knowledge is introduced in the models of the IASs.	Focus group, Wizard-of-Oz study, Data mining, Heuristic evaluation, Play with layer, Simulated users, Cross-validation
4. Deciding upon adaptation (DA)	Determining the necessity of as well as the required type of, adaptations by the IAS given a particular state of the "world". The particular state is expressed in the various models maintained by the system.	Focus group, Wizard-of-Oz study, Heuristic evaluation, Cognitive walkthrough, Simulated users, Play with layer, User test
5. Applying (or instantiating) adaptation (AA)	Instantiating the actual adaptations in the user–system interaction.	Focus group, Wizard-of-Oz study, Heuristic evaluation, Cognitive walkthrough, User test, Play with layer
6. Evaluating adaptation as a whole	Evaluate the overall adaptation	Heuristic evaluation, Cognitive walkthrough, User test , Play with layer
7. All layers	Evaluate the entire system	Focus group , Cognitive walkthrough, Heuristic evaluation, User test

Our research focused on generalizing an existing model that provides non-adaptive dialogues for conceptual database design to provide adaptive support in multiple domains. The existing model's ability to provide dialogue support has already been validated empirically (Weerasinghe & Mitrovic, 2006), and therefore the first three layers have already been evaluated. As the result, only the last four adaptation layers are relevant for this research.

Now we discuss some of the evaluation methods related to the last four layers.

1. **Focus group:** This is a type of interview conducted with groups. Participants are expected to provide their opinion on issues in an informal group setting, facilitated by a moderator.

2. **Wizard-of-OZ study**: This method provides a structured way for using humans to inspire the algorithms needed in an adaptive system. The wizard is typically an expert, who performs the functions that the system should perform (before implementation). The users (or students in this case) used the system typically without knowing that a human is involved.

3. **Heuristic evaluation:** In principle this method can be applied to every layer, as long as the suitable heuristics are agreed upon. This method involves a usability expert judging the focused layer against a set of criteria. The experts need to have expertise in heuristic evaluation, need to understand the meaning of the particular heuristics and questions used, and need to understand the layer's input and output.

4. **Cognitive walkthrough**: This method focuses on usability. This technique involves usability experts working through typical user tasks, and deciding for each action whether a novice user might encounter difficulties.

5. **Simulated users:** Testing computer systems with real users generally tend to be costly in both financial and temporal terms. This situation is even worse when evaluating an adaptive system due to the difficulties of getting users to participate long enough for the adaptation to be fully tested. Using simulated users involve computational methods of users instead of real users. The advantage of using simulated users is that different aspects of adaptation can be tested rapidly, and that the system inputs for the different layers can be controlled.

6. **User test:** This method can be used when the implementation of the functionality corresponding to an evaluation layer is complete. Typically, users are given well-defined tasks to

do. Measurements are made of users' performance (e.g., how fast they learn in an ITS) and opinions.

7. **Play with layer:** This is a type of user test in which participants are not given tasks, but allowed to freely explore the layer. They freely input data as if coming from the preceding layer in the adaptation process, and judge the layer's behaviour. Judging can involve a pre-specified criteria or objective measures such as frequency of occurrences of certain events, such as adaptations. Questionnaires or interviews can also be used to obtain participants' opinions.

Now we discuss the methods that were used in our research. We need to evaluate two aspects of our model:

(i) generality of our model in providing tutorial dialogue support (discussed in Chapter 4). This refers to applying (or instantiating) adaptation, 5th layer of the framework for evaluating IASs (Table 3.2). The suitable evaluation methods are focus group, Wizard-of-Oz study, heuristic evaluation, cognitive walkthrough, user test and play with layer. We used Wizard-of-Oz study as further discussed in Section 4.1.

(ii) effectiveness of the adaptation of tutorial dialogues in enhancing learning (discussed in Chapter 4). This refers to evaluating all the layers of an IAS, the last adaptation layer of the framework. The suitable evaluation methods are focus group, cognitive walkthrough, heuristic evaluation, and user test. We used user test as further discussed in Section 4.2.

3.3 Model for adaptive tutorial dialogue support

For my M.Sc. thesis, KERMIT was extended to facilitate dialogue support in learning conceptual database design. For an erroneous student solution with multiple errors, the model selects the pedagogically most suitable error for discussion and initiates the corresponding dialogue. The dialogue discusses the error in the context of the current problem as well as the relevant domain concept. However there are two limitations. The first one was the dialogue support was limited to a single domain, conceptual database design. The other was dialogues provided by the model were non-adaptive. i.e. two students with different knowledge and different tutoring histories receive the same dialogue when identical solutions are submitted.

In this research, we generalised the existing model in two ways: (i) Dialogue support was provided for multiple domains (ii) Dialogue support was made adaptive based on both the student model and the history of the tutoring session. Facilitating dialogue support in multiple domains was achieved by generalising the model. In order to achieve (ii), we developed adaptation rules to customise the dialogues for each individual student.

The generalised model consists of three components: an error hierarchy, a set of tutorial dialogues and rules for adapting them. We now discuss each of these components.

3.3.1 Error hierarchy

The error hierarchy categorizes all error types in a domain. At the lowest level, an error type is associated with one or more violated constraints, which forms leaves of the hierarchy. The error types are then grouped into higher level categories. When there are multiple errors in a student solution, the hierarchy is used to select the pedagogically most suitable error for discussion.

Starting from the previous error hierarchy developed for KERMIT, we explored how we can generalise the hierarchy to categorise errors represented in EER-Tutor (a web-based reimplementation of KERMIT) and NORMIT constraint bases. The domain knowledge of EER-Tutor is represented as a set of 212 constraints (Mitrovic et al., 2007), whereas NORMIT consists of 82 constraints (Mitrovic et al., 2012). We now discuss how this investigation is used to generalise the error hierarchy. We then focus on how the student model is extended based on the error hierarchy.

Structure of the error hierarchy: Figure 3.2 represents a high-level view of the hierarchy showing the top three levels. This hierarchy has gone through several revisions and the final version is presented here.

As mentioned before, an incorrect solution violates one or more domain constraints. Incorrect knowledge for a domain is much greater than correct knowledge and leads to violation of one or more constraints for that domain. The error hierarchy divides the space of the incorrect knowledge into a set of sub-areas: *Basic syntax errors* and *Errors dealing with the main problem-solving activity*. Each sub area is further divided as presented in Figure 3.2. Under the new node *Basic syntax errors*, we included simple syntax errors such as checking whether the student has filled the required fields, the components used to fill the required fields are valid and so on. Hence it is

sufficient to discuss such errors using a single message. The other category requires a dialogue to be conducted.

ALL ERRORS
- Basic syntax errors
- Errors dealing with the main problem-solving activity
  - Using an incorrect solution component type
  - Extra solution component types
  - Missing solution component types
  - Associations
  - Failure to complete related changes

Figure 3.2: High-level view of the final version of the refined error hierarchy

In the previous hierarchy developed for KERMIT, the space of incorrect knowledge was divided into Syntax errors and Semantic errors. Introducing these two nodes (*Basic syntax errors* and *Errors dealing with the main problem-solving activity)* was necessary because of the situations in which it was not sufficient to present a single feedback message for some violated syntax constraints in the NORMIT constraint base: a dialogue was required. In other words, the errors categorized under the node *Basic syntax errors* form a subset of syntax errors for which only a single feedback message is sufficient.

Figure 3.2 presents the domain-independent part of the error hierarchy. Further categorisations of the nodes presented in the high-level view focus on the domain specific details. For example, consider the detailed view of the node *Using an incorrect solution component type* for conceptual database design (Figure 3.3) and data normalization (Figure 3.4). In both these cases, each of the nodes categoriesd under the node *Using an incorrect solution component type* focuses on domain-specific details. The numbers in Figures 3.3 and 3.4 specify the constraints that are assigned to the leaves of the error hierarchy. Even though the constraints are independent of each other, the error hierarchy groups the constraints that are related to a single error type. Some error types are associated with a single constraint, whereas some correspond to multiple constraints. For instance, the error type *Using a regular relationship to represent an identifying relationship* is associated with a single constraint 28_1 (Figure 3.3). On the other hand, constraints 27 and 28 are grouped under the node *Using an entity to represent a relationship* (Figure 3.3). This is because these two constraints correspond to different errors that the students can make. Constraint 27 focuses on the

Using an incorrect solution component type
- ER Constructs
  - Using a completely different type of solution component
    - **Using an entity to represent another type of solution component**
      - Using an entity to represent a relationship
        - 27 or 28
      - Using an entity to represent an attribute
        - 202 or 203 or 204 or 205 or 206 or 202-1or 205-1or 206-1
    - **Using a relationship to represent another type of solution component**
      - Using a relationship to represent an entity
        - 13_1or 14_1
      - Using a relationship to represent an attribute
        - 207 or 208 or 209 or 210 or 211_A or 207_1 or 210_1 or 211_B
    - **Using an attribute to represent another type of solution component**
      - Using an attribute to represent an entity
        - 13_2 or 14_2
        - 65-5
        - 65-6
      - Using an attribute to represent a relationship
        - 27_2 or 28-2
  - Using a different variation of the correct solution component
    - **Entity**
      - Using a regular entity to represent a weak entity
        - 14
        - 67-2
      - Using a weak entity to represent a regular entity
        - 13
    - **Relationship**
      - Using a regular relationship to represent an identifying relationship
        - 28_1
      - Using an identifying relationship to represent a regular relationship
        - 27-1
    - **Attribute**
      - Using a different type of complete attribute
        - 54 or 54_1 or 55 or 56 or 57 or 57_1 or 58 or 58_1 or 59 or 59_1
        - 65-2
      - Using a different type of component attribute
        - 41 or 42
- EER Constructs
  - Using a completely different type of solution component
    - **Sub class**
      - Using a weak entity to represent a sub class
        - 115
    - **Category**
      - Using a weak entity to represent a category
        - 130
  - Using a different variation of the correct solution component
    - **Sub class**
      - Sub class doubling as a category
        - 113
    - **Category**
      - Category doubling as a sub class
        - 128

Figure 3.3: Detailed view of the node *Using an incorrect solution component type*

situation when a regular relationship is modelled incorrectly using an entity, whereas the other deals with an identifying relationship.

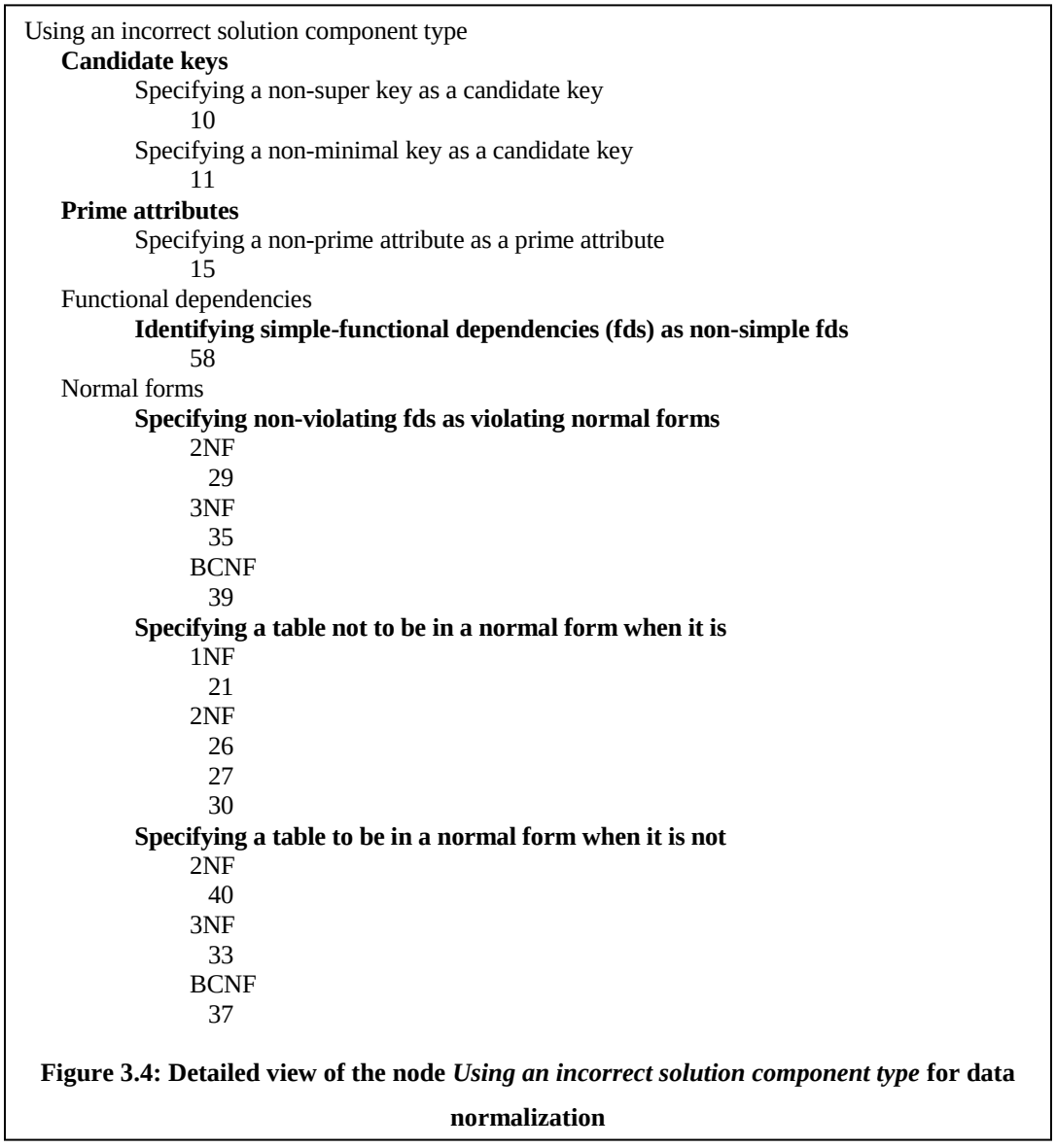


Figure 3.4: Detailed view of the node *Using an incorrect solution component type* for data normalization

The node *Using an incorrect solution component type* categorises all the errors that deal with a solution component being modelled as another type. For example, the node *Using a different type of a complete attribute* (a child node of the *Using an incorrect solution component type)*, categorises errors that occur when the wrong type of attribute is used to model an attribute. i.e.

when a key attribute is modelled as a simple attribute. A similar example from the hierarchy for data normalization is the node *Specifying a non-prime attribute as a prime-attribute*.

As can be expected, the size of the sub-tree for the node *Using an incorrect solution component type* is different for the two domains. This sub-tree for conceptual database design has 8 levels whereas the sub-tree for data normalization has 7 levels. The complete error hierarchies for conceptual database design and data normalization are presented in Appendices A.1 and A.2 respectively.

While the errors categorized under the node *Using an incorrect solution component type* focus on components that should be part of the correct solution but were modelled incorrectly. the node *Extra solution component types* focuses on components that should not be part of the correct solution. An example from conceptual database design can be an extra entity that should not be included in the solution (Figure A.5 in Appendix A.1). In the case of data normalization, it can be a trivial functional dependency that should not be part of the solution (Figure A.25 in Appendix A.2).

As the name suggests, the node *Missing solution component types* categorise errors that deal with components that should be part of the solution but have not been included. Errors caused by a missing regular entity (Figure A.6 in Appendix A.1) or a missing candidate key (Figure A.26 in Appendix A.2) are examples of errors that can be categorised under the node *Missing Solution component types.*

The node *Associations* categorizes all errors dealing with various types of associations between different solution components. For instance, errors focusing on relationships between entities, entities and attributes and relationships and attributes belong to this node. This node was added to replace two nodes: *Connecting an attribute to an incorrect construct* and *Errors dealing with cardinalities and participation* (Figure 2.23) that were specific to the conceptual database design. As these two previous nodes deal with relationships between solution components, it is appropriate to replace them with the new node.

A new node *Failure to complete related changes* has been added. This node focuses on students' inability to complete related changes that are needed as a result of other changes that they have carried out. The constraints for this new node focus on the subsequent changes triggered by a change in a student solution. These constraints are known as retrospective constraints and they are

not part of the existing constraint-bases that we have explored. This is based on an observation from a Wizard-of-Oz study (Section 3.3.3): some students seemed to be reacting to feedback on errors by making suggested changes without reflecting on other modifications that also needed to be carried out. As an illustration, in conceptual database design if a regular entity with a key attribute is changed to a weak entity, a partial key should be specified instead of the key attribute. This may increase the number of required attempts to arrive at the correct solution, leading to frustration.

The names of some nodes on the highest level were changed to make the hierarchy more general. For instance, *Extra constructs* was renamed to be *Extra solution component types, Missing constructs* to *Missing solution component types* and *Using an incorrect construct type* to *Using an incorrect solution component type* (Figure 3.2).

Pedagogically significant nodes: The error hierarchy structures the space of incorrect knowledge of an instructional domain. If we track the errors a student makes in terms of the error hierarchy, then that information could be used to drive important pedagogical decisions such as selecting the most suitable error to discuss when a solution has multiple errors. Constraint-based tutors typically use a student's performance on a single constraint to select the most suitable error. However, we hypothesise that it is more effective to consider a student's performance on each error type instead on individual constraints. However some error types at the higher levels of the hierarchy are not discriminative enough as they cover multiple error types. For instance, the error types at the top three levels of the hierarchy cover a wide range of errors and are not suitable to provide a detailed view of a student's performance. Therefore, we identified a set of pedagogically significant nodes useful for selecting the most suitable error for discussion. Identification of these nodes is based on our experience on teaching database courses, developing database tutors and advice from a domain expert. In conceptual database design, the node *Using an entity to represent another type of solution component* (Figure 3.3) is identified as a pedagogically significant node as it covers errors that occur when an entity is used to represent a relationship or an attribute. However, this node's parent *Using a completely different type of solution component* is not considered as pedagogically significant node because it covers three types of errors (i.e. using an entity to represent another

type of solution component, using a relationship to represent another type of solution component and using an attribute to represent another type of solution component).

We present the pedagogically significant nodes in bold for the sub tree *Using an incorrect solution component type* in conceptual database design (Figure 3.3). Similarly these significant nodes are given in bold for data normalization in Figure 3.4.

We now discuss how we extended the student model to record information about mistakes made.

Extending the student model: The student model is extended with the information about the errors the student made during interaction. This new component of the student model stores the frequency of making a mistake for each node of the error hierarchy. One way to calculate the frequency of making a mistake f_i for each constraint i as follows.

$$f_i = \frac{Number\ of\ times\ the\ constraint\ has\ been\ violated}{Number\ of\ times\ the\ constraint\ has\ been\ relevant}$$

This calculation includes the student's entire history of using that constraint, and does not take into account that earlier attempts at applying the constraint may no longer be valid. A better method is to consider the last x times the constraint has been relevant. Then x can be used instead of the total number of times the constraint is relevant. In the case of total being less than x, the total should be used. This allows the influence of previous attempts to be controlled. The choice of x will clearly have an effect on the calculations. For example, choosing a value too large will result in knowledge of a constraint being influenced by irrelevant history, but a value too small will disregard previous attempts that are still relevant. We chose x to be five as it has been previously used successfully for similar purposes in SQL-Tutor (Mitrovic, 2012)

Calculating the frequency of making an error for higher level nodes: Each higher-level node has one or more constraints as descendants. Frequency of making an error for a higher level node can be calculated as a combination of the frequencies of each individual constraints belonging to that node. A simple heuristic is to calculate the average of the frequencies of the individual

constraints. However, a modification is required to allow for the possibility that some or all of the constraints belonging to a non-leaf node have not been relevant at the time of calculation.

Consider a higher-level node *k* which has *n* number of constraints as descendants. Let f_i be the frequency of violating constraint *i*, *y* be the number of constraints belonging to node *k* for which frequencies are unknown and f_k be the frequency of making a mistake at the non-leaf node. Two options were considered for calculating f_k.

1. If $y > 0$ then f_k = unknown

   else $f_k = \frac{\sum_{i=0}^{n} fi}{n}$

2. If $y = n$ then f_n = unknown

   else $f_n = \frac{\sum_{i=0}^{n} fi}{n-y}$

Option 1 avoids the problem of unknown frequencies by reporting a frequency for a non-leaf node only when the frequency for each of the constraints belonging to that node are known. Otherwise the frequency is said to be unknown. This causes problems when some constraints are not relevant for some problems. For example, the non-leaf node *Entity*, a child node of ***Extra constructs*** has three constraints. Two of them deal with weak entities whereas the other one focuses on regular entities. Even though every ER model contains at least one regular entity, weak entities are not part of some ER models that students are expected to develop. Consequently, the frequency for the non-leaf node *Entity* could not be calculated until a student has used the system long enough to see such a problem which requires at least one weak entity in the ER model. To overcome this, we used option 2, which relaxes the condition that all constraint frequencies probabilities are required. It does this by including only the known frequencies when calculating the average frequency of making a mistake for each non-leaf node. This provides an estimate of how frequently an error is made at each non-leaf node.

The advantage of this heuristic is that it can be easily used in a dynamic environment, where the student models are continually changing.

3.3.2 Tutorial dialogues

Error remediation is facilitated through tutorial dialogues. A dialogue is designed for each error type (i.e. each leaf node in the hierarchy). Prompting students about their mistakes is taking advantage of a good learning opportunity as prior research suggests that successful learning occurs when students make an error (Ohlsson, 1996; VanLehn, Siler, Murray, & Baggett, 1998).

There are two types of dialogues: (i) Single-level dialogues and (ii) Multi-level dialogues. Single-level dialogues handle errors associated with simple syntax errors for which a detailed feedback message is sufficient to explain the error. Hence these dialogues are limited to a single feedback message. An example of a single level dialogue is *You have connected an entity A to entity B directly. Entities cannot be directly connected to each other*. These are similar to single-turn dialogue moves supported by AutoTutor (Person et al., 2001). Multi-level dialogues handle more complex errors for which a series of prompts is necessary to guide students to self-explain both domain concepts and solution steps. These dialogues consist of four levels:

1. Conceptual prompt: informs the student about the corresponding domain concept that he/she has difficulty with and asks the student about the particular aspect of the domain concept,
2. Reflective prompt: prompts the student to understand why his/her problem-solving action is incorrect,
3. Corrective action prompt: prompts the student to specify the correct problem-solving action and
4. Conceptual reinforcement prompt: helps the student to review the domain concept learnt via a question.

Reflective and the corrective action prompts discuss the error within the context of the current problem. In contrast, the other two prompts (conceptual and conceptual reinforcement prompts) focus on the particular aspect of the domain concept associated with the error. As mentioned in Section 2.4, there are two exceptions to the structure of multi-level dialogues. The former refers to those dialogues that discuss errors related to cardinality and participation; they consist of only three levels. The second exception is the different focus of the reflective prompt when focusing on missing constructs. Both these exceptions will be discussed in detail at the end of this section.

We now discuss each prompt using the dialogues presented in Figures 3.5 and 3.6. The hypothetical dialogue in Figure 3.5 is initiated when a regular entity is used to represent a weak entity, an error in conceptual database design. The complete dialogue is given in Appendix B.1.

EER-Tutor1: You seem to be having some difficulty with regular entities. Can you tell me the general rule to decide whether something is a regular entity?
Student1: *I'm not sure*

EER-Tutor2: Let's try a simple thing. What is used to uniquely identify instances of a regular entity?
Student2: partial key attribute

EER-Tutor3: That's not quite right. A key attribute is used to uniquely identify instances of a regular entity.
EER-Tutor4: Now tell me why modelling *CHAPTER* as a regular entity is incorrect?
Student3: *I'm not sure*

EER-Tutor5: Let's see …Check carefully whether you can find a key attribute for *CHAPTER*.
{After *Continue* button is pressed}
Now please select the key attribute for *CHAPTER*.
Student4: *No*

EER-Tutor6: Good job! So now can you tell me best way to model *CHAPTER*?
Student5: weak entity

EER-Tutor7: Well done! Now use the correct word to fill in the blanks so that you'll remember what you learnt.
A …….. uniquely identify instances of a …….
Student8: key attribute, regular entity

EER-Tutor8: Great job! Please make the changes and submit the solution when you're done.

Figure 3.5: A Hypothetical dialogue in EER-Tutor

Figure 3.6 discusses an error in data normalization. It is initiated when the specified candidate key is incorrect because it has extra attributes and is not a minimal key. The complete dialogue is given in Appendix B.2.

Each prompt is associated with three possible options. One of the options is correct and one is incorrect. Options such as "I need more help", "I'm not sure" provide the opportunity for students to specify that they do not know the correct answer and ask for assistance.

The first level, the conceptual prompt, specifies the domain concept relevant to the most frequently made error type and then asks the student for a particular aspect of the corresponding domain concept. For example, "You seem to be having some difficulty with regular entities. Let's look at regular entities in detail. Can you tell me the general rule to decide whether something is a regular entity?" (EER-Tutor1 in Figure 3.5) is an example of a conceptual prompt. Here the

concept that the student has difficulty with is regular entities. This concept is covered in several constraints. This dialogue is associated with one particular aspect of the domain concept: i.e. the general rule to decide whether something is a regular entity. Explicitly indicating the domain concept that students have difficulty with provides an opportunity to reflect on them. Even though the first level prompt focuses on the conceptual prompt, the offered options are not strict definitions from the textbook. Instead the student needs to reason about how corresponding domain concept is related to current problem-solving action. This prompt does not reduce the task to recognition, but requires the student to re-examine his/her domain knowledge in order to answer the question. If the student fails to select the correct answer, he/she is given a simpler question. In this dialogues the simpler question used is "What is used to uniquely identify instances of a regular entity?" (EER-Tutor2 in Figure 3.5). If the student still has difficulty answering the simpler question, the system provides the correct answer. A correct answer by a student at any stage moves the dialogue to the next stage and the next prompt is displayed.

The reflective prompt, the second level prompt provides an opportunity to reflect on why the current problem-solving action is incorrect. This prompt requires the use of both procedural and conceptual knowledge. In contrast to the conceptual prompt, the reflective prompt discusses the selected error within the current problem-solving context. For instance, "Tell me why modelling *CHAPTER* as a regular entity is incorrect?" (EER-Tutor4 in Figure 3.5) is an example of a reflective prompt in this dialogue. If a student fails to answer the reflective prompt correctly (as shown by Student3 in Figure 3.5), he/she will be given assistance to understand why the current problems-solving action is incorrect (EER-Tutor5 in Figure 3.5). Here the student is expected to understand that CHAPTER should not be modelled as a regular entity because CHAPTER does not have a primary key attribute.

The third level prompt, the corrective action prompt assists the student to understand how to correct the mistake. The prompt "Can you tell me the best way to model CHAPTER?" (EER-Tutor6 in Figure 3.5)" is an example of a corrective action prompt in this dialogue.

The conceptual reinforcement prompt, the final prompt is another opportunity to learn the corresponding domain knowledge. The prompt presented as EER-Tutor7 in Figure 3.5 helps the student to understand that a key attribute uniquely identifies instances of a regular entity.

We developed 31 single-level and 112 multi-level dialogues and these dialogues cover all the errors in conceptual database design. Thus we showed that the proposed structure is expressive enough to discuss all the errors in conceptual database design.

We used the same structure to develop dialogues for data normalization. Based on the hierarchy we developed 33 single-level and 45 multi-level dialogues. A hypothetical dialogue in Figure 3.6 is presented as an example.

NORMIT1: You seem to be having some difficulty with candidate keys. Let's look at candidate keys in detail. What is a candidate key?
Student1: *I'm not sure*

NORMIT2: Let's try a simple thing. What do we mean by a candidate key being a minimal set of attributes?
Student2: *A candidate key: is the smallest set of prime attributes that determines all the attributes*

NORMIT3: That's not quite right. A candidate key: is the smallest set of attributes that determines all the attributes.
NORMIT4: Now can you tell me what is wrong with AD as a candidate key?
Student3: *I'm not sure*

NORMIT5: Let's see …Check carefully whether all the attributes of AD are needed to determine all the other attributes.
{After *Continue* button is pressed}
Do we need all the attributes of AD to determine all the other attributes?
Student4: *No*

NORMIT6: Good job! We need only some of the attributes of AD to determine all the other attributes. So now tell me the extra attribute(s) for AD?
Student5: *D*

NORMIT7: Well done! Now try to answer this question so that you'll remember what you learnt.
A ..... is a .....key with extra .....
Student8: superkey, candidate key, attributes

NORMIT8: Great job! Please make the changes and submit the solution when you're done.

Figure 3.6: A hypothetical dialogue in NORMIT

This dialogue is initiated when the specified candidate key is incorrect because it is not a minimal key i.e. the specified candidate key has extra attributes. If the specified candidate key is AD (one of the correct candidate keys is A) for step 1 in the example presented in Data Normalization (Section 3.1.3), then it is a non-minimal key (i.e. super key). This is because of the extra attribute D in the specified candidate key. This error violates constraint number 11 which is

categorised under the node *Specifying a non-minimal key as a candidate key*, which is a child of the node *Candidate keys* (Figure 3.5).

The conceptual prompt (NORMIT1 in Figure 3.6) focuses on candidate keys, the corresponding domain concept for the current error. If a student fails to answer this prompt correctly, he/she is given simpler question (NORMIT2 in Figure 3.6). Failure to answer the simpler question results in presenting the correct answer by the tutor. Providing the correct explanation at any stage, moves the dialogue to the next stage and the next prompt is displayed.

The reflective prompt "Now can you tell me what is wrong with AD as a candidate key?" (NORMIT4 in Figure 3.6) assists the student to reflect on why the current-problem solving action is incorrect. Failure to answer this prompt correctly, results in a prompt that guides the student to reflect on the current-problem solving action (NORMIT5 in Figure 3.6). In this dialogue, the tutor helps the student to understand that all the attributes in the specified candidate key are not needed to determine all the other attributes in the given relations. The corrective action prompt provides an opportunity to understand how to correct the mistake (NORMIT6 in Figure 3.6). Here the tutor helps student to clarify the extra attributes in the specified candidate key. Finally the conceptual reinforcement prompt provides another opportunity to learn the corresponding domain concept (NORMIT7 in Figure 3.6).

Dialogues with three levels: As mentioned earlier, dialogues that discuss cardinality and participation consist of only three levels. This is because there are the only two possible values for cardinality (i.e. 1 or N). These are the only ones allowed by the interface (Figure 2.2). As soon as the student is made aware that the cardinality specified is incorrect, the correct answer becomes obvious. Asking students to specify the cardinality using another prompt might potentially demotivate them if they want to resume problem solving straightway. When they correctly explain why the specified cardinality is incorrect, they are given the final prompt of the dialogue. For example, the response will be "Great Job! I guess you know how to correct the mistake now. Before starting to make changes, try to answer this question. What is the correct question to ask when deciding the participation between entities E1 and E2 in a binary relationship?". Similarly participation also has two possible values: total and partial. As soon as the ITS tells the student that the specified participation is incorrect, then the correct answer becomes apparent. Thus the

dialogues that discuss errors related to participation also have only three levels. There are four dialogues that focus on cardinality and four dialogues that discuss errors related to participation.

Different level-2 prompt when discussing missing constructs: The reflective prompt focuses on why the current problem-solving action is incorrect. However, this is not applicable when the error is about missing solution components. In such situations, the student is asked to specify the type of construct that is missing.

Our paper-based investigations focus on developing tutorial dialogues for two different types of instructional tasks: conceptual database design and data normalization. Thus we showed that the proposed structure is expressive enough to discuss all the errors in both domains.

3.3.3 Adaptation rules

A human tutor's ability to adapt to a student's evolving knowledge is a key factor contributing to the effectiveness of human one-on-one tutoring. Researchers try to develop ITSs that can achieve the same effectiveness. However, there is no consensus on how to develop ITSs that closely approximate human tutors as they follow a variety of strategies. Observing human tutors in a teaching and learning environment is a common method used to understand how adaptation could be supported in an ITS. We conducted a Wizard-of-Oz study to observe how human tutors interact with students while they use EER-Tutor. We now discuss the study, its findings and how they were used to design the adaptation rules, the final component of our model. The adaptation rules are used to customise the dialogues based on an individual student's tutoring session and the student model. These rules were designed and developed as they did not exist in the previous model developed for my M.Sc. research.

Data gathering for adaptation rules − The Wizard-of-Oz study: The study involved observing human tutors interacting with students while they learn with EER-Tutor. The study was conducted at the University of Canterbury, and involved student volunteers enrolled in an introductory database course and experienced human tutors. All human tutors had several years of tutoring experience providing assistance on request to students in labs and/or teaching small groups. As

EER-Tutor complements classroom instruction, the study was scheduled after the relevant learning material was taught in the classroom.

The version of EER-Tutor used in the study was enhanced with a chat interface (Figure 3.7), so that the human tutor could provide one-on-one feedback to students. Even though we were planning to replace the typical feedback from the ITS with the tutorial dialogues, we still allowed the participants to receive typical feedback. This is to compensate for the delay in presenting feedback through the chat interface due to the time required for typing. In addition, we wanted to make the bandwidth between the student and the human tutor very similar to that between the student and the ITS. Therefore, participants interacted with the system in one room and the human tutors observed their interactions in another room.

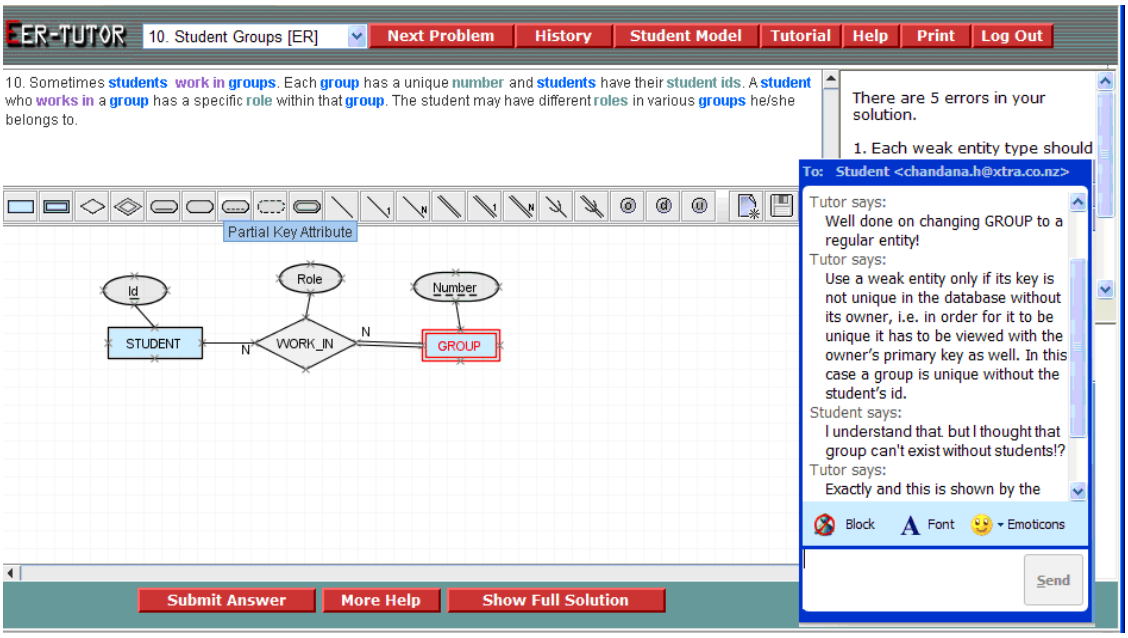


Figure 3.7: Interface of the enhanced version of EER-Tutor

As this study was conducted as a data gathering exercise, human tutors were not given any specific instructions on providing assistance. They were asked to guide the participants towards solutions using appropriate methods like asking questions, providing explanations etc. Participants were not told that a human tutor was involved in the study. They also had the opportunity to initiate intervention through the chat interface or using the *More Help* button in the interface

(Figure 3.7). The participants were free to end the session whenever they wanted. All learner interactions were recorded.

The participants were asked to fill out a questionnaire at the end of the session to understand their perceptions about the system and interventions through the chat interface. At the end of each session, the human tutors were also interviewed to understand their views on the tutoring experience.

We initially analysed the recordings without the human tutors, to investigate how students were prompted by different human tutors and which interactions triggered these prompts. In the second phase, whenever possible, the recordings were discussed with the human tutors to clarify how they decided on the timing and the level of feedback provided through the chat interface.

Even though Wizard-of-OZ studies are a common method to gather details about human-one-on-one tutoring, this experimental set up varies from previous studies of tutorial dialogues in a number of ways. First, the human tutors in this study provided support in addition to the feedback given by the system. The human tutors also responded to learners' questions. This contrasts with those studies of Chi et al. (2001) and Grasser, Person and Magliano (1995), in which the human tutor is expected to lead the dialogue through a series of questions. Second, the learner interacts both with the system and the human tutor. Although Merill et al. (1992) have studied tutorial dialogues in the context of problem-solving, the human tutor was the only source of feedback for the student as he/she solved problems on paper.

Results and analysis: Seven students and four human tutors participated in the study, with at most two students per tutor. The average duration of the sessions was 85 minutes (sd=20). The average number of problems attempted was 11 (sd = 5) and all participants completed all attempted problems. We discuss results in two different categories: (i) type of feedback provided in the interventions and (ii) timing of interventions.

Type of feedback provided: We analysed the interactions between the human tutors and the students in order to identify dialogues, each pertaining to a single topic. There were a total of 69 dialogues. The maximum and the minimum number of dialogues initiated by a human tutor during a session was 20 and 4 respectively. These 20 dialogues occurred in a session of 1.5 hour duration.

In the session which consisted of only 4 dialogues, the first dialogue occurred only in the 19th problem (the student completed 22 problems). In addition to discussing the current error or the relevant domain concept, some of the dialogues focused on helping with the interface (such as labelling constructs), motivating and praising the student, suggesting the student try a more challenging problem or helping with technical problems (e.g. web browser suddenly closing).

We are mainly interested in 37 dialogues which discussed the current problem state or the relevant domain concepts. The following statistics were calculated for these 37 dialogues. The average number of such dialogues per human tutor was 9.25. Five dialogues contained a single utterance each, which was initiated by the human tutor. For instance, a human tutor utterance that occurred just after the completion of a problem was "*Remember that the participation for weak entity is always total*". The longest dialogue consisted of nine utterances of which four were by the human tutor. The student made more utterances than the human tutor in only two dialogues. Furthermore, only two dialogues were student-initiated. This indicates that the human tutor is more likely to be active in the interventions.

An example is presented in Figure 3.8, which occurred while a student was solving the problem in Figure 3.7. In this dialogue, the student was able to identify and repair the misconception he had with weak entities and total participation. The highest number of dialogues in a session was seven, while the lowest was two. As can be expected, the highest number of dialogues occurred in the longest session.

Tutor1:	Well done on changing GROUP to a regular entity!
Tutor2:	Use a weak entity only if its key is not unique in the database without its owner, i.e. in order for it to be unique it has to be viewed with the owner's key as well. In this case a group is unique without the student's id.
Student1:	I understand that, but I thought that group can't exist without students!?
Tutor3:	Exactly, and this is shown by the total participation in the relationship**.**
Student2:	I have learned something today

Figure 3.8: Example of a self-explanation dialogue

When data was analysed to identify different techniques used by human tutors, three techniques were prominent: (i) Tutors were rephrasing feedback from the ITS, (ii) providing problem-independent explanations and (iii) stating the tutor's observations before starting to discuss the problem state. The first technique, rephrasing feedback enables the student to

understand their own mistake. For example, the human tutor prompted "Does AUTHOR need to be an entity?" or "The cardinalities of BORROWED_FROM needed fixing". Rephrasing feedback may have been effective because most students realised that the additional feedback was provided by a human tutor observing their problem-solving process. If our model is to repeat the same kind of prompting, it is difficult to ascertain whether it will have the same effect (Lepper, Woolverton, Mumme, & Gurtner, 1993). The second technique was to discuss the current problem state and then provide a problem independent explanation (Tutor2 in Figure 3.8). These explanations provided an opportunity for the student to repair his mental model of the domain and generated further conversation. The third technique was to state the human tutor's observations before starting to discuss the problem state. For example, human tutor started the dialogue by saying "You seem to be having a few problems with relationships. Think about this. Can a student be enrolled in a course without involving a department?"

Learning of constraints: As the knowledge base in EER-Tutor is represented as a set of constraints, the errors were recorded as violations of constraints (Mitrovic et al., 2007). We analysed how frequently constraints were violated after related errors were discussed using dialogues, to see whether tutor interventions helped students to improve their knowledge. If these constraints represent psychologically appropriate units of knowledge, then learning should follow a smooth curve when plotted in terms of constraints (Anderson, 1993). To evaluate this expectation, the participants' logs were analysed, and each problem-state after a tutor invention in which a constraint was relevant was identified. These identified occasions are referred to as *occasions of application*. Each constraint relevance occasion was ranked 1 to *n*. For each occasion we recorded whether a relevant constraint was satisfied or violated. We then calculated the frequency of violating a constraint on the first occasion of application, the second occasion and so on, for each participant. The frequencies were then averaged across all participants and plotted as a function of the number of occasions when a constraint was relevant (Figure 3.9). At the first occasion, all participants had some relevant constraints, whereas only two participants had a constraint relevant at n = 17. This indicates that the number of constraints that were relevant decreases as occasion number increases.

As can be seen from Fig. 3.9a, there is an outlier, increasing the frequency of violating a constraint in the 4th and the 5th occasions. This is due to a single student violating the constraint dealing with total participation of entities. For this student, the human tutor provided a problem-independent explanation on total participation to help him identify and repair the misconception he had with weak entities and total participation (Figure 3.8). The explanation was not related to a problem state later on, as the discussion was a follow-up from another error related to weak entities. This may have been a reason for the subsequent violations of this constraint.

Figure 3.9b shows the learning curve with the outlier removed. The frequency of 0.22 for violating a constraint at its first occasion of application decreased to 0.02 at its eighth occasion of application, displaying a 90.9% decrease in the frequency. The results of the mastery of constraints reveal that students seem to learn ER modelling concepts which were discussed by the human tutors.

Twenty-eight different constraints were discussed in the dialogues. Three students did not violate any constraints in subsequent occasions after the human tutor interventions. These students were tutored by three different human tutors who followed techniques like rephrasing feedback, providing problem-independent explanations and stating tutor's observations at the beginning of the discussion. This suggests that at least some of these techniques have been effective in helping the students learn domain concepts.

Timing of Interventions: The original version of EER-TUTOR provides feedback on demand, i.e. only when the student submits the solution. In addition to the feedback from the system, the human tutors in this study provided delayed feedback, which was well received by the participants. Delayed feedback also provided an opportunity for students to correct the mistakes themselves. There were a few instances where the student made a mistake and corrected it after referring to the problem text again. For example, one of the problems required students to model CAR as an entity and *Colour* as a multi-valued attribute of CAR. The student modelled *Colour* as a simple attribute and then changed it to multi-valued as the last sentence in the problem text indicated that a car can have many colours. In such a situation, immediate feedback would not have been welcomed by the student as he/she may have felt the intervention intrusive.

The important issue with delayed feedback is how the human tutors decided that the students needed help as they were not given any specific instructions on tutoring. In our study human tutors provided help when the student (i) made the same type of mistake repeatedly; (ii) asked for more help using the *More Help* button; (iii) was inactive for some time; (iv) reacted to feedback or (v) asked a reflective question through the chat interface.

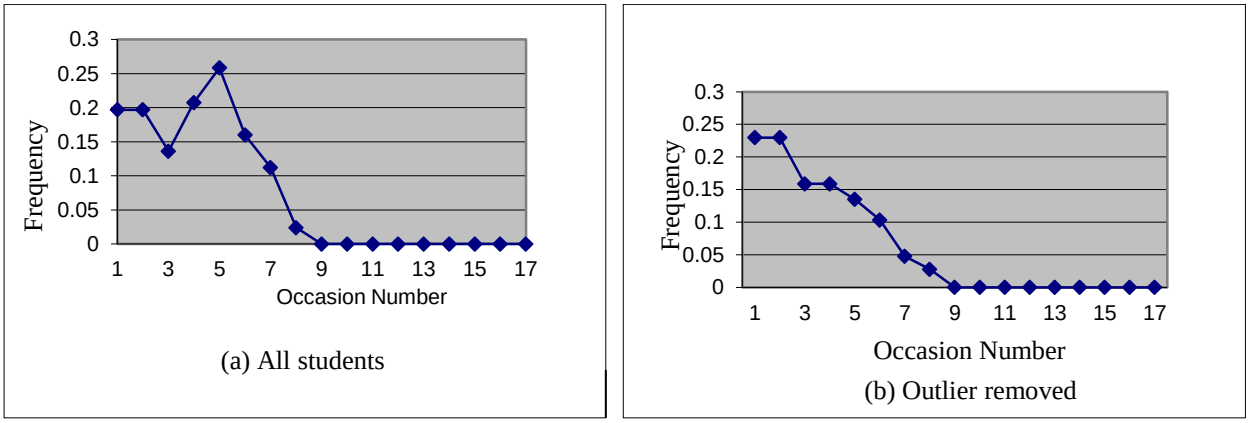


Figure 3.9: Probability of violating a constraint

Questionnaire Responses: This section summarises the responses to the user questionnaire which all participants were asked to complete at the end of the study. The questionnaire had 18 questions, out of which eight focused on human tutor interventions**.** The remaining questions focused on enjoyment of the system, amount learnt etc. Only three questions were completely close-ended i.e. responses were limited to a set of given options. Seven questions consisted of a close-ended part (i.e. responses were limited to a rating on a Likert scale or a pre-specified set of options) followed by an open-ended section on which the participants could comment as necessary. The remaining eight questions were open-ended providing the participants an opportunity to respond as they wanted. Table 3.3 presents the mean responses for the questions that had a Likert scale.

The response to the question related to previous experience with conceptual database design indicated that 86% (6 out of 7) of the volunteers had participated in lectures and engaged in some additional learning. Lectures had been the only source of learning for only one participant. Even though the previous experience indicated is approximately the same, the individual knowledge level can be different.

We now discuss the students' responses about the human tutor interventions. The participants indicated that the human tutor interventions were helpful in identifying their mistakes (mean responses was 4.5 in Table 3.3) and were easy to understand (mean responses was 4.57 in Table 3.3). All participants except one (this participant indicated he/she did not know) indicated that interventions helped them to learn conceptual database design concepts (question 9 of the questionnaire). Therefore the user responses indicated that the dialogue prompts helped them to reflect on their own mistakes, learn domain concepts and were easy to understand.

Table 3.3: Mean responses from the questionnaire Wizard-of-Oz study 1

	No of responses	**1 to 5 on Likert scale**	**mean**	**s.d.**
How helpful were human tutor interventions to identify your mistakes on your own	6	"Not at all" to "Very much"	4.5	0.84
How easy were the human tutor interventions to understand	7	"Not at all" to "Very easy"	4.57	0.79
Amount learnt (includes both help through the chat interface and the typical EER-Tutor feedback)	7	"Nothing" to "Very much"	4.29	0.76
Enjoyment	7	"Not at all" to "Very much"	4.57	0.78
Ease of using interface	7	"Not at all" to "Very easy"	4.0	0

When asked about the timing of the interventions through the chat interface (question 8(a) of the questionnaire), 86% of the participants indicated that the help was given at the right time. Only one participant indicated that the help was given earlier than needed. All the participants indicated help was given at the right level of detail (question 8(b) of the questionnaire). They appreciated that the help provided just enough detail to be guided towards the correct solution rather than the exact action to correct a mistake.

Three questions addressed different aspects such as the enjoyment, ease of using the interface and amount learnt (Table 3.3). When asked how much they learnt about conceptual database design (which includes both the typical feedback by the system and the help through the chat interface), the mean response was 4.29. When asked about the time needed to learn the system's functions, 71% (5 out of 7) of the participants indicated that they needed less than 5 minutes. One participant needed 10 minutes whereas the remaining one 30 minutes. This is also consistent with

them indicating that the interface was easy to use with a mean response of 4.0. They also indicated that they enjoyed learning with EER-Tutor with a mean response of 4.57. Furthermore all the participants indicated that they would recommend EER-Tutor to others (question 5 of the questionnaire).

Discussion: Even though this is a limited study involving only seven participants, analysis of learning constraints reveals that students learned the ER modelling concepts which were discussed by the human tutors. Findings from the study were used to design the adaptation rules. We will now discuss each rule in detail.

Design of adaptation rules: Adaptation rules enable individualization of the dialogues, based on the student model. The rules decide on the timing, selection and content of each dialogue for each individual student. These rules have gone through several revisions and the current version is presented in Table 3.4.

Feedback provided in a problem-solving environment teaches how to correct the current mistake, as well as the relevant conceptual knowledge. There is no consensus among ITS researchers about different issues related to providing feedback: (i) focus (which error to focus on when there are multiple errors in the current attempt); (ii) timing (on demand, delayed or immediate); and (iii) content (information about correcting the selected mistake vs. relevant conceptual knowledge) (Koedinger & Aleven, 2007; VanLehn, 2011; Mitrovic, 2012). The way the issue (i) is addressed by most dialogue-based systems is utilising the history of the tutoring session to decide which error to focus on (Aleven, Popescu, et al., 2003; Evans & Michael, 2006; VanLehn et al., 2007). However, we hypothesise that it is more effective to consider not only the history of the tutoring session but also the evolving knowledge of the student. This novel approach of combining the history of the tutoring session with the student's knowledge provides a more detailed view of the student that is useful for the ITS to drive pedagogical decisions. Rule 1 solely focuses on using the student's knowledge, whereas rules 2 and 3 also use the history of the tutoring session.

Table 3.4: Adaptation Rules

Rule Identifier	Rule
1	**Selecting the dialogue** IF SS is incorrect THEN Select the dialogue for the error type that was most frequently made

2	**Selecting the starting prompt of the chosen dialogue**
	IF same mistake has been repeated 3 times or more within the same session THEN
	Start the explanation from conceptual prompt of the chosen dialogue
	ELSE
	Start the explanation from Reflective prompt of the chosen dialogue

3	**Selecting the ending prompt of the chosen dialogue**
	IF current prompt is the conceptual prompt THEN
	Move to the next prompt
	ELSE IF ((student is seeing the current prompt for the first time in the session AND the response for the current prompt is correct) OR the current prompt is the last prompt in the dialogue) THEN
	Allow the student to resume problem-solving
	ELSE
	Move to the next prompt

4	**Inactivity time-long**
	IF SS has not been changed for the last 8 minutes and has not been evaluated at least once THEN
	Evaluate SS
	Tell the student that they have been inactive and ask whether any assistance is needed
	
	IF student confirms he/she needs assistance THEN
	Initiate the chosen dialogue

5	**Inactivity time-short**
	IF SS has not being changed for the last 4 minutes and has been evaluated at least once previously THEN
	Evaluate SS
	Tell the student that they have been inactive and ask whether any assistance is needed
	
	IF student confirms he/she needs assistance THEN
	Initiate the chosen dialogue

6	**Abandoning a problem**
	IF student has changed a problem without completing (for well-defined tasks, abandoning includes changing the problem after completing an intermediate step) THEN
	Evaluate SS
	Inform the student that he has not completed the problem, and ask whether he/she wants to change the problem or want help
	
	IF student confirms he/she needs assistance THEN
	Initiate the chosen dialogue

7	**Abandoning problems in succession**
	IF student has changed 3 problems consecutively without completing them (for well-defined tasks, abandoning includes changing the problem after completing an intermediate step) THEN
	Evaluate SS
	State that three problems has been abandoned consecutively
	Ask whether he/she wants to change the problem or want help
	
	IF student confirms he/she needs assistance THEN
	Initiate the chosen dialogue

Some dialogue-based systems provide feedback on demand (Mitrovic, 2005; Weerasinghe & Mitrovic, 2006), while the others provide feedback immediately (Aleven, Koedinger, & Popescu, 2003; Evans & Michael, 2006; VanLehn et al., 2007). In the former case, learner is given more control, whereas the system takes more control in the latter. Constraint-based tutors generally provide feedback on-demand. However, our adaptation rules support both ways of providing feedback. In addition to providing on-demand feedback when a student submits a solution, the model provides immediate feedback in some pre-specified situations. When a student abandons the current problem (i.e. requests a new problem without completing the current one) or has been inactive for a period of time, the model asks the student whether he/she needs assistance. Upon conformation, the model provides feedback. Even though some students might need immediate assistance in these situations, some prefer to take more time to work through the problem themselves. Therefore, the student is still given control and responsibility to decide whether he/she needs assistance. Rules 4 and 5 cover situations of the student being inactive for a pre-specified time period, whereas rules 6 and 7 focus on abandoning of problems.

Some dialogue-based systems separate teaching the correct problem-solving action from the relevant conceptual knowledge (VanLehn et al., 2007; Mitrovic, 2005), while others combine them (Evans & Michael, 2006; Weerasinghe & Mitrovic, 2006; Heffernan et al., 2008). Our tutorial dialogues use the combined approach. They not only assist the student to understand the correct problem-solving action for the current context but also help him/her to learn the corresponding domain concept. Our approach is to focus on multiple aspects (reflecting on the current error, discussing the correct problem-solving action, discussing corresponding domain concept and reviewing the domain concept) of the current error. This approach makes it possible for our dialogues to facilitate knowledge construction as well as knowledge remediation. Even though dialogues provided by our model have multiple prompts focusing on different aspects of the current error, the entire dialogue is not presented each time an error is made. The number of prompts provided increases with the number of occurrences of the error and the accuracy of the answer. e.g. for the first occurrence of an error, only the reflective prompt is presented. The second occurrence of the same error triggers both the reflective prompt and the corrective-action prompt. In addition, failure to answer any prompt triggers the subsequent prompt of the dialogue. This approach of increasing the level of detail in relation to the occurrences of an error type is

similar to increasing the information content of each feedback message with each subsequent request for assistance (Anderson et al., 1996; Koedinger & Aleven, 2007; Mitrovic, 2012). Rules 2 and 3 focus on this adaptation of the dialogues. We now discuss each rule in detail.

Rule 1 addresses the critical issue of selecting a dialogue. Dialogue selection is very important because if it is not effective, it might be difficult for students to acquire deep knowledge of the domain. Multiple errors indicate a student's difficulty in understanding multiple domain concepts. Selecting a single error from a set of errors to focus on is known as *error localization* (Heffernan et al., 2008). Discussing a single error assists the student to focus on a single aspect of his/her solution during a complicated problem-solving process.

In our model, the selection of the pedagogically most suitable error for discussion is based on the constraint violations of the last five attempts. The algorithm used to select the best error is presented in Figure 3.10. Based on our domain expertise, a set of higher-node levels have been pre-specified as pedagogically significant, enabling us to identify the higher-level error type that was most frequently made (as discussed in Section 3.3.1). After finding the corresponding error type (i.e. pedagogically significant higher-level node) for each violated constraint, we will focus on the error type that was most frequently made.(i.e. the pedagogically significant node with the highest frequency.) However, there are two exceptions. The first one occurs when at least one of the violated constraints corresponds to a basic syntax error. Such errors are very simple and easy to correct and sometime it is not possible to make progress towards the correct solution without correcting these errors. Thus constraints that belong to the node *Basic Syntax errors* are given precedence. If there is more than one pedagogically significant node that belongs to the node *Basic Syntax errors*, the one with the highest frequency is chosen (step 4 Figure 3.10). The second exception occurs when the most frequently made error type is associated with missing solution components (i.e. a child of the sub-tree *Missing solution components*). Focusing on missing solution components instead of the incorrect ones results in giving away a lot of information rather than letting the students to attempt to solve the problem. Hence we then look for pedagogically significant nodes that are not categorised under the node *Missing solution components* and select the one with the highest frequency (steps 7, 8 and 9 in Figure 3.10). The errors that are categorized under *Missing solution components* are considered only if they are the only errors remaining to be discussed (step 12 in Figure 3.10).

1. PS ← Find the set of pedagogically significant higher-level nodes corresponding to all violated constraints
2. BS ← Find the subset of PS categorised under the node Basic Syntax errors
3. IF BS subset is not empty
4. BS_{max} ← Select the node with highest frequency from BS
5. RETURN the highest frequency leaf-node that is a descendant of BS_{max} and contains a violated constraint
6. ELSE
7. NMS ← Find the subset of higher level nodes that are not categorised under Missing solution components
8. IF NMS subset is not empty
9. NMS_{max} ← Select the node with highest frequency from NMS
10. RETURN the highest frequency leaf-node that is a descendant of NMS_{max} and contains a violated constraint
11. ELSE
12. PS_{max} ← Select the node with highest frequency from PS
13. RETURN the highest frequency leaf-node that is a descendant of PS_{max} and contains a violated constraint

Figure 3.10: Algorithm to select the pedagogically most suitable error for discussion

Once the pedagogically significant node with the highest frequency is selected, we then select its child which has the highest frequency that contains a violated constraint. We repeatedly select the child with the highest frequency until the leaf node with a violated constraint is selected. The descendant is identified as the constraint to focus on. This approach takes into account both the current (violated constraints of the current problem) and the recent behaviour related to constraints. However, it does not focus on the entire history of constraints as the last five attempts could potentially provide a more realistic view of the student's current knowledge level. This approach not only considers the learning of each constraint, but also the knowledge of the overall domain to decide the best error to discuss.

Rules 2 and 3 customise the content of the selected dialogue and are based on the human tutors' use of conceptual and reflective explanations during the Wizard-of-Oz study with EER-Tutor. Rule 2 customises the starting prompt of the dialogue, whereas rule 3 focuses on the ending prompt. If a mistake is done for the first time within a session, the dialogue starts with the reflective prompt aiming to assist the student to reflect on the error within the current context. This prompts helps the student to focus on the corresponding conceptual knowledge as well as the procedural knowledge for the current error. For example, the reflective prompt when a weak entity

is modelled as a regular entity is *Why is modelling CHAPTER as a regular entity incorrect?* (EER-Tutor4 in Figure 3.5). To answer correctly, the student needs to know the difference between weak and regular entity types (i.e. A regular entity type has a key attribute whereas a weak entity type does not) as well as whether any of attributes for CHAPTER can be a key attribute. Thus the reflective prompt helps the student to focus on both conceptual as well as procedural knowledge for the current error.

However, if the mistake is repeated for three times or more, we believe that there is some evidence of lack of knowledge of the corresponding domain concept. We use a heuristic here, which has been used successfully in other projects by our group (Mitrovic, 2012). When the same error is repeated three times or more, the dialogue focuses on the corresponding conceptual knowledge before moving on to the reflective prompt. Before focusing on the conceptual knowledge, the model will state the higher level error-type most frequently made as an observation (EER-Tutor1 in Figure 3.5). This is one of the strategies human tutors used in the Wizard-of-Oz study. Prompting the students to focus on the corresponding domain concept only when the same mistake is repeated at least three times rather than every single time is in line with the dialogues' role to facilitate error remediation. Similarly, we do not expect the students to go through the entire dialogue to understand a mistake in their solution. Therefore the student is encouraged to resume problem-solving when the student demonstrates that he/she understood what is being discussed. This understanding is monitored by the accuracy of the response for the dialogue prompt. i.e. if the student has answered the reflective prompt for an error he has made for the first time in the current session, then he is asked to resume problem-solving. However, if the student's subsequent problem-solving behaviour indicates that he/she still has difficulty with the domain concept discussed by the dialogue, then the level of detail provided by the dialogue is increased. For instance, the first occurrence of an error triggers the reflective prompt of the corresponding dialogue (Rule 2). As a correct response to that prompt provides evidence that the student has understood why his problem-solving action is incorrect, he/she is encouraged to resume problem-solving. Every time a mistake is repeated within a single session, the student receives one more level of the dialogue than the previous instance before being able to resume problem-solving. i.e. a dialogue for a specific error keeps expanding with one more level for each occurrence of that error with one exception. When an error has occurred three times, the dialogue will expand by two

levels: (i) the expansion just discussed (ii) the dialogue will start at the conceptual prompt. In other words, three or more occurrences of a mistake in a single session trigger the entire dialogue to be delivered to the student. Hence the ending prompt of the dialogue (dealt by Rule 3) is based on the correctness of the current response and the number of occurrences of the mistake within a single session.

When a new session starts, dialogues start presenting only the reflective prompt when an error occurs for the first time. Thus customisation of the dialogues is based on the student's explanation skills (as evidenced by the response to the dialogue prompts) and the subsequent problem-solving behaviour within a session.

As Table 3.5 indicates, the student who answers the prompts correctly receives the minimum number of prompts. On the other hand, a student who answers each prompt incorrectly is expected to go through the entire dialogue to understand why what he/she did was wrong.

Table 3.5: Minimum and maximum number of prompts received by a student

No of occurrences of an error	Initial prompt	Subsequent prompts received by a student	
		For each correct response	For each incorrect response
1	Reflective prompt	Resume problem solving	Corrective action prompt Conceptual reinforcement prompt
2	Reflective prompt	Corrective action prompt	Corrective action prompt Conceptual reinforcement prompt
3 or more	Conceptual prompt	Reflective prompt Corrective action prompt Conceptual reinforcement prompt	Reflective prompt Corrective action prompt Conceptual reinforcement prompt

Rules 4 and 5 deal with a student's inactivity while interacting with the ITS. The inactivity might be due to different reasons such as being stuck, confused, bored or being in a reflective mode. These rules evaluate the student solution after a pre-specified time period of inactivity has elapsed. These system-initiated evaluations are called *silent evaluations*. As the evaluation of the student solution after a period of inactivity can potentially provide important insights to the knowledge level of the student, the error hierarchy will be updated. After updating the error hierarchy, the user is requested to confirm that he/she needs assistance. Giving the opportunity to confirm needing assistance may lessen the annoyance the user may feel being given assistance

when not required. If the student indicates that he/she needs assistance, the chosen dialogue will be initiated. Rule 4 focuses on the situation in which he/she has not yet received any feedback on the current problem. This might be due to the time required at the beginning of the problem-solving process to be familiar with the problem. On the other hand, rule 5 focuses on the situation when the student has received feedback at least once for the current problem. As we do not want to be intrusive, a longer time period has been specified when the student is still familiarising him/herself with the problem (as covered by rule 4) than the one in which he/she has received feedback at least once (covered by rule 5). Even though it was difficult to specify the ideal time period of inactivity, the times were specified based on our experience teaching an introductory database course.

Rules 6 and 7 focus on when a student abandons a problem without completing it. Rule 6 is triggered when a single problem is abandoned, whereas rule 7 is activated when three problems are abandoned consecutively. Abandoning a problem may occur at three different stages of solving a problem: (i) prior to start developing a solution (i.e. solution is empty) (ii) while developing a solution but before receiving feedback at least once and (iii) while developing a solution after receiving feedback at least once. A problem may be abandoned due to several reasons: (i) problem being too easy (ii) problem being too difficult (iii) wanting to attempt a different problem (Mitrovic, 2001)

Abandoning a problem triggers a silent evaluation and an update of the error hierarchy. Updating the error hierarchy enables the model to use the important information related to a student's knowledge level when a decision to abandon a problem is made. Then the user is requested to confirm that he/she needs assistance. Upon conformation the chosen dialogue is initiated.

As the adaptation rules do not depend on domain specific details to individualise dialogues, they can be used across domains.

3.4 Discussion

As can be expected, the size of the error hierarchy (i.e. number of levels and the number of nodes) depends on the size of the constraint base and grouping of constraints to form higher-level error

types. For instance, the sub-tree *Using an incorrect solution component type* for conceptual database design consists of eight levels, whereas it is seven levels for data normalization.

One of the important decisions taken by our model is the selection of the pedagogically most suitable error for discussion. This is based on a student's demonstrated ability for each domain concept. Error selection in existing dialogue-based systems is based on the history of the tutoring session (VanLehn et al., 2000; Aleven, Popescu, et al., 2003; Evans & Michael, 2006). In contrast, our model keeps track of a student's ability to apply each domain concept since he/she starts using the particular constraint-based tutor, but considers only the more relevant short-term view for error selection (i.e. a student's ability to apply a domain concept is based on the last five opportunities to apply that constraint). Thus selection of the most suitable error is based on their recent behavior related to the constraints.

The error hierarchy can also be used to understand how specific each constraint is. Even though each constraint should identify a specific type of error, some constraints may be violated due to more than one incorrect problem-solving step. For instance, several constraints that need to be made more specific were identified during the exploration of the data normalization domain. For example, when simplifying functional dependencies (fds) one of the initial checks was to compare the number of fds in the correct solution with that of the student solution. This constraint can be violated either by having too many or too few fds in the student solution. As a result, this constraint can be categorized under both *Missing solution components* or *Extra solution components*, leading to two different dialogues. Therefore, this constraint needs to be made more specific so that it deals with only one type of error.

Our tutorial dialogues not only assist the student to understand the correct problem-solving action for the current context, but also help him/her learn the corresponding domain concept. Thus our dialogues are focused not only on a student's current performance but also on his/her future performance. A student's current performance focuses on the correct problem-solving action for the selected error. On the other hand, the future performance is associated with learning the domain concept associated with the error reducing the likelihood of repeating the same error. Our approach is to focus on multiple aspects (reflecting on the current error, discussing the correct problem-solving action, discussing corresponding domain concept and reviewing the domain concept) of

the selected error. This approach makes it possible for our dialogues to facilitate knowledge construction as well as knowledge remediation.

We have attempted to maximize learning efficiency by expecting students to go through just one dialogue prompt for the first occurrence of an error. The number of prompts a student receives for a certain type of error depends on the number of occurrences of that error as well as the accuracy of the student's explanations. Learning efficiency is further enhanced by focusing on the error that a student is most likely to make in subsequent attempts.

Chapter 4

Evaluation

Evaluation is an integral part of research. We believe that the effectiveness of an educational system can only be determined through empirical evaluations with real users.

We now need to validate the model's ability to provide adaptive dialogue support in multiple domains. As discussed in Section 3.2, we need to evaluate two different aspects of the model:

(i) the generality of our model in providing tutorial dialogue support

(ii) the effectiveness of the adaptation of tutorial dialogues in enhancing learning

Some evaluations were carried out during the development of the model, while others were conducted after the full implementation. Section 4.1 reports on the evaluations performed during the development process, while Section 4.2 is dedicated to the latter.

4.1 Development-based evaluations of the model

In order to evaluate the generality of our model in providing tutorial dialogue support (the objective (i) above), we conducted paper-based investigations involving two other domains: logical database design and fraction addition (Section 4.1.1). Each of these domains has a well-defined domain theory and is a well-defined instructional task. In order to achieve the effectiveness of the adaptation of tutorial dialogues in enhancing learning (objective (ii)), the prototype was used to provide adaptive dialogue support for ERM-Tutor, a constraint-based tutor that teaches logical database design, in a Wizard-of-OZ study (Section 4.1.2).

4.1.1 Evaluating the generality of the model

Previously, we used two domains: conceptual database design and data normalization (Section 3.3). Conceptual database design is an ill-defined instructional task, whereas data normalization is well-defined (as discussed in Section 3.1.3). In this section, we discuss how we evaluated the

generality of the model with two other domains, logical database design and fraction addition. Logical database design involves mapping high-level, conceptual ER schemas to relational schemas using the 7-step mapping algorithm (Elmasri & Navathe, 2010). Both logical database design and fraction addition have well-defined domain theories and are well-defined instructional tasks. Even though fraction addition domain is very simple, it is quite different from the other three domains (conceptual database design, logical database design and data normalization) that all relate to the database design process that we have investigated previously.

First we explored whether the domain-independent part of the error hierarchy (i.e. top three levels of the error hierarchy presented in Figure 3.2) can be applied in these two domains. Second we explored whether the proposed dialogue structure (Section 3.3.2) can be applied in these two domains. Further details are given below. Adaptation rules, the last component of the model was evaluated in a Wizard-of-Oz study (Section 4.1.2).

Evaluating the error hierarchy: In order to evaluate the generality of the error hierarchy, we developed the error hierarchy for both logical database design and fraction addition. First we discuss the error hierarchy in logical database design that was developed using the constraint base of ERM-Tutor. We started with the domain-independent part of the error hierarchy and extended it further to assign all the 82 constraints of the ERM-Tutor (Milik et al., 2006). 27 constraints were categorised under the node *Basic syntax Errors* and 55 constraints were categorised under the *Errors dealing with the main problem-solving activity*. The detailed view of the node *Using an incorrect solution component type* for logical database design is presented in Figure 4.1. The first digit of the constraint identifier indicates the step of the mapping algorithm in which the constraint is applicable. For instance, constraint 217 (a child node of the node *Using a non-primary key as a foreign key*) is applicable in step 2 of the algorithm.

As can be seen from Figure 4.1, the same type of error can occur in different steps of the algorithm. For instance, a student might use an attribute that is not the primary key as a foreign key. This error can occur when a weak entity (step 2) or a 1:1 relationship (step 3) is mapped. On the other hand, the same type of error can occur for different types of constructs mapped within a single step. For example, both constraints 314 and 315 are associated with step 3 of the algorithm. These constraints focus on mapping an attribute that is not the primary key as a foreign key.

Constraint 314 covers the scenario when the foreign key is associated with a regular entity, whereas constraint 315 deals with a foreign key associated with a weak entity.

Using an incorrect solution component type
- Using a weak entity instead of a regular entity
  - 103
- Using a regular entity instead of a weak entity
  - 204
- Using a regular relationship instead of an identifying relationship
  - 206
- Using an identifying relationship instead of a regular relationship
  - 303, 405, 503, 704
- Using a binary relationship instead of a non-binary relationship
  - 304, 403, 504
- Using a non 1:1 relationship instead of a 1:1 relationship
  - 305
- Using a non 1:N relationship instead of 1:N relationship
  - 404
- Using a 1:N relationship instead of a M:N relationship
  - 505
- Using a non n-ary relationship instead of a n-ary relationship
  - 703
- Using a composite attribute instead of its components
  - 106, 210
- Using a non-candidate key as a candidate key
  - 114
- Using a non-primary key as a foreign key
  - 217, 314, 315, 414, 415, 511, 512, 606, 710, 711
- Using a multi-valued attribute instead of a non multi-valued attribute
  - 603

Figure 4.1: Detailed view of the node *Using an incorrect solution component type for logical database design*

The node *Missing constructs* form the largest sub tree with four levels, whereas all the other nodes form sub trees with three levels. The complete error hierarchy for logical database design domain is presented in Appendix A.3. The top three-levels of this error hierarchy are domain-independent and identical to those indicated in Figure 3.2. Therefore we were able to use the domain-independent parts of the error hierarchy to develop an error hierarchy for the logical database domain.

We repeated the process of developing an error hierarchy for the domain of fraction addition. The task is well-defined with a well-defined domain theory. The constraint base generated by ASPIRE (Mitrovic et al., 2009), an authoring tool to develop constraint-based tutors was used. Of the 32 constraints, 21 constraints were categorised under the node *Basic syntax Errors* and 11 constraints were categorised under *Errors dealing with the main problem-solving activity*. The detailed view of the node *Using an incorrect solution component type* for fraction addition is presented in Figure 4.2.

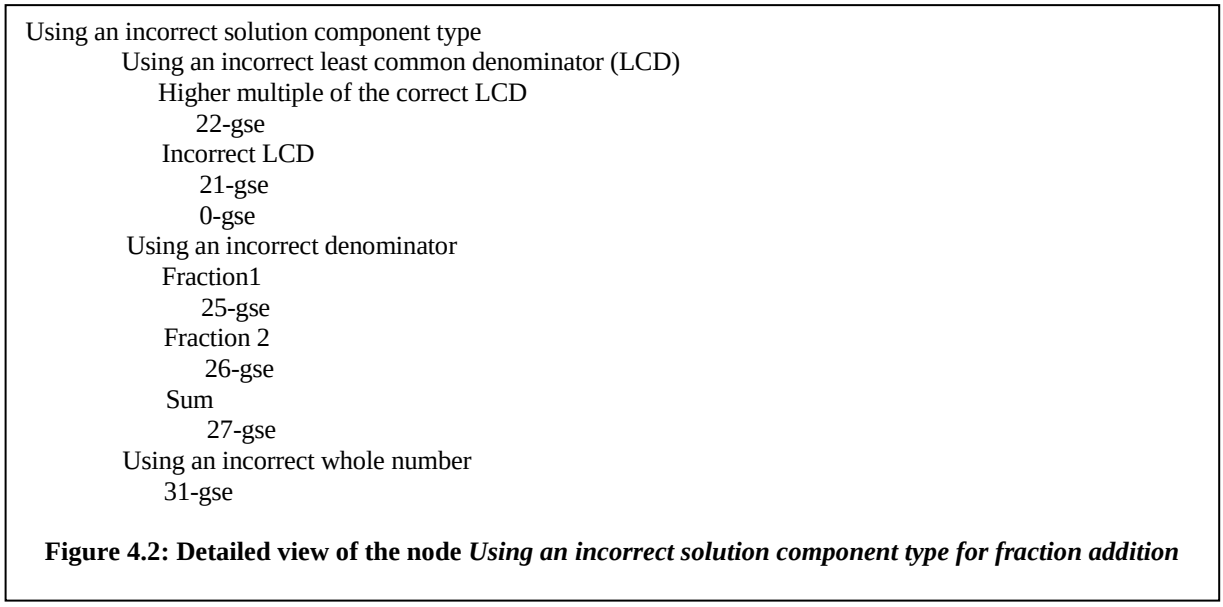


Figure 4.2: Detailed view of the node *Using an incorrect solution component type for fraction addition*

The node *Basic syntax Errors* form the largest sub tree with 5 levels whereas each of the other nodes (*Using an incorrect component type*, *Extra constructs*, *Missing constructs*, *Associations*) form the smallest sub tree with three levels. The complete error hierarchy for fraction addition domain is presented in Appendix A.4. The top three levels of this error hierarchy are domain-independent and identical to those indicated in Figure 3.2. Therefore we were able to use the domain-independent parts of the error hierarchy to develop an error hierarchy for the fraction addition domain.

We have now discussed four paper-based investigations involving four domains of different complexities: conceptual database design, data normalization, logical database design and fraction addition. Thus we have provided evidence that we were able to develop an error hierarchy for each

domain starting from the domain-independent parts of the hierarchy. Each of the hierarchies categorises all error types for that domain.

Evaluating tutorial dialogues: In order to evaluate the generality of the proposed dialogue structure in other domains, we explored the same two domains: logical database design and fraction addition. First we explored the domain of logical database design. We now illustrate how the error "using an attribute that is not a primary key as a foreign key" occurs in this domain and the corresponding dialogue.

Logical database design is the process of converting the conceptual schema (represented as an ER model) to the implementation schema, using the 7-atep algorithm. The logical schema presented in Figure 4.3 is based on the ER diagram given in Figure 2.7(b). Here we consider the step that involves mapping weak entities (step 2 of the mapping algorithm). That means the step 1 that focuses on mapping regular entities is already completed. i.e. mapping of the regular entity TEXT_BOOK is already complete. Now we focus on mapping the weak entity CHAPTER. The student solution has one error: an attribute that is not the primary key (i.e. *Name)* is specified as a foreign key. The foreign key that should be included in CHAPTER is ISBN, which is the primary key of the owner entity (i.e. TEXT_BOOK for this scenario).

(a) Student Solution
TEXT_BOOK(ISBN, Name)
CHAPTER (Chapter_Number, Name, Topic, Number_of_pages, Number_of_references)

(b) Ideal Solution
TEXT_BOOK(ISBN, Name)
CHAPTER (Chapter_Number, ISBN, Topic, Number_of_pages, Number_of_references)

Figure 4.3: A student's solution (a) and the ideal solution (b)

Figure 4.4 represents a hypothetical dialogue that discusses this error, corresponding to the node *Using a non-key attribute as a foreign key*, which is a child node of *Using an incorrect solution component type* (Figure 4.1) of the error hierarchy. The complete dialogue is presented in Appendix B.3.

This dialogue helps a student understand the concept of foreign keys (ERM-Tutor1 in Figure 4.4). It also assists the student to understand that he/she needs to focus on the concept of foreign keys as he/she seem to find it difficult (based on the frequency of errors made). This is an example of a conceptual prompt. If the student fails to answer the conceptual prompt correctly, he/she is presented with a simpler prompt (ERM-Tutor2 in Figure 4.4). If the student is still unsuccessful in answering the prompt successfully, then the correct answer is presented. A correct answer at any level moves the dialogue to the next level and the next prompt is displayed.

ERM-Tutor1:	You seem to be having some difficulty with foreign keys. Let's look at foreign keys in detail. What is a foreign key?
Student1:	*I'm not sure*
ERM-Tutor2:	Let's try a simple thing. How do we make a connection between weak entity A and its owner entity B?
Student2:	Including an attribute of A in B
ERM-Tutor3:	That's not quite right. We make a connection between weak entity A and its owner entity B by including the primary key of B in A.
ERM-Tutor4:	Now tell me why including Name as a foreign key is incorrect?
Student3:	*I'm not sure*
ERM-Tutor5:	Let's see …Check carefully whether Name is the primary key attribute for *TEXT_BOOK*? {After *Continue* button is pressed} Now please tell me whether *Name* is the primary key attribute for *TEXT_BOOK*?
Student4:	*No*
ERM-Tutor6:	Good job! Now can you tell me the foreign key to be included in *CHAPTER*?
Student5:	*Isbn*
ERM-Tutor7:	Well done! Now use the correct word to fill in the blanks so that you'll remember what you learnt. When mapping a weak entity …….key is the …….key of the owner entity.
Student6:	*foreign, primary*
ERM-Tutor8:	Great job! Please make the changes and submit the solution when you're done

Figure 4.4: Hypothetical dialogue in ERM-Tutor

The next prompt presented is the reflective prompt as indicated by "Now tell me why including Name as a foreign key is incorrect?" (ERM-Tutor4 in Figure 4.4). It provides an opportunity to reflect on why the student's problem-solving action is incorrect. If the student's

attempt is unsuccessful, then he/she is given assistance to understand why his/her problem-solving action is incorrect (ERM-Tutor5 in Figure 4.4).

The next prompt also known as the corrective action prompt assists the student to focus on how to correct the error (ERM-Tutor6 in Figure 4.4). Finally the conceptual reinforcement prompt as indicated by ERM-Tutor7 (Figure 4.4) provides another opportunity to focus on the corresponding domain concept.

We developed 17 single-level and 48 multi-level dialogues and these dialogues cover all the errors in logical database design. Thus we showed that the proposed structure is expressive enough to discuss all the errors in logical database design.

Next we explored the applicability of the proposed dialogue structure in the domain of fraction addition. Figure 4.5 represents a hypothetical dialogue that discusses the least common denominator (LCD) for two given integers being incorrect. i.e. the correct LCD of 6 and 8 is 24, but the student's answer is 12. This error is categorised under the node *incorrect LCD*, which is a child node of *Using an incorrect LCD* (Figure 4.2). The complete dialogue is presented in Appendix B.4.

MathTutor1:	You seem to be having some difficulty calculating the least common denominator (LCD). Let's look at LCD in detail. Can you tell me why we need the LCD?
Student1:	*I'm not sure*
MathTutor2:	Let's try a simple thing. Do you know what the least common denominator (LCD) mean?
Student2:	*The smallest positive integer that divides the given integers without a remainder*
MathTutor3:	That's not quite right**.** LCD is the smallest positive integer that is a multiple of the given integers.
MathTutor4:	Now tell me why having 12 as the LCD of 6 and 8 is incorrect?
Student3:	*I'm not sure*
MathTutor5:	Let's see …Check carefully whether 12 is a multiple of both 6 and 8. If 12 is a multiple of 6, we can say 12 = 6 * 2? {After *Continue* button is pressed} Now can you tell me what you know about 12 being a multiple of both 6 and 8?
Student4:	*12 is a multiple of 6, but 12 is not a multiple of 8.*
MathTutor6:	Good job! Now tell me the correct LCD in this case?
Student5:	*24*
MathTutor7:	Well done! Now use the correct word to fill in the blanks so that you'll remember what you learnt. …….. is used to convert the …….. of the fractions to be added to a common ……..
Student6:	*LCD, denominators, denominator*
MathTutor8:	Great job! Please make the changes and submit the solution when you're done.

Figure 4.5: A hypothetical dialogue in the fraction addition domain

The first prompt, known as the conceptual prompt helps a student understand the concept of LCD (MathTutor1 in Figure4.5). It also emphasizes the fact that the student needs to focus on LCD as it is a difficult concept for him/her based on the student model. Failure to answer this prompt correctly, results in a simpler conceptual prompt (MathTutor2 in Figure 4.5). Failing to answer the prompt correctly results in receiving the correct answer by the system. A correct answer at any level moves the dialogue to the next level and the next prompt is displayed.

The next prompt presented is the reflective prompt as indicated by the prompt MathTutor4 (Figure 4.5). It provides an opportunity to reflect on why the student's problem-solving action is incorrect. In this dialogue, the reflective prompt assists the student to understand why 12 is not the LCD of 6 and 8. If the student's attempt is unsuccessful, then he/she is given assistance to understand why the problem-solving action is incorrect (MathTutor5 in Figure 4.5).

The next prompt, known as the corrective action prompt assists the student to focus on how to correct the error (MathTutor6 in Figure 4.5). Finally the conceptual reinforcement prompt as indicated by MathTutor7 (Figure 4.5) provides another opportunity to focus on the corresponding domain concept.

We developed 21 single-level and 11 multi-level dialogues and these dialogues cover all the errors in fraction addition. Thus we showed that the proposed structure is expressive enough to discuss all the errors in fraction addition.

We have now discussed how we developed tutorial dialogues in four domains of different complexities: conceptual database design, data normalization, logical database design and fraction addition. Each of these domains is associated with a well-defined domain theory. However, conceptual database design is the only ill-defined task, all the others are well-defined. Thus we have provided evidence that the proposed structure is expressive enough to discuss all the errors in each of the four domains.

4.1.2 Evaluating the adaptation of tutorial dialogues

Our investigations indicated that the first two components of the model, the top three levels of the error hierarchy and the dialogue structure, could be used in the domains of conceptual database design, logical database design, data normalization and fraction addition. The final component of the model, adaptation rules, were developed based on the findings of the Wizard-of-Oz study

involving EER-Tutor (Section 3.3.3). Prior to the full implementation of the model, a second Wizard-of-Oz study was conducted to evaluate the effectiveness of the prototype of the model. As the first Wizard-of-Oz study involved teaching conceptual database design, an ill-defined task with a well-defined domain theory, we chose a different type of instructional task for the second study. Therefore the second study involved ERM-Tutor that teaches logical database design, a well-defined task with a well-defined domain theory.

Testing the effectiveness of the adaptation rules refers to applying or instantiating adaptation, 5th layer of the framework for evaluation of IASs (Table 3.2). The suitable evaluation methods are focus group, Wizard-of-Oz study, heuristic evaluation, cognitive walkthrough, user test and play with layer. If the task under evaluation is one that humans may be good at, then focus groups and Wizard-of-Oz studies are suitable. As the gold standard for ITSs is expert human tutoring, we chose Wizard-of-Oz study as the evaluation method to evaluate the adaptation rules. In this study, we observed how human tutors interact with students while they learn with ERM-Tutor. We now present the study and its findings.

Experimental setup: The study involved student volunteers and experienced tutors and was conducted at the University of Canterbury. Two types of feedback were provided: typical feedback provided by the system, and dialogues initiated by the model. The behaviour of the model was simulated by the author. Even though the full implementation of the model would replace the typical ITS feedback, we decided to keep this feedback due to the anticipated delay in the manual process of simulating the actions of the proposed model. The dialogue support was provided through a chat interface (see Figure 4.6), and will be referred to as interventions hereinafter. As it was very difficult to provide these interventions in a timely fashion due to the time required to type the prompts, the dialogue support was provided only when an error was repeated two or more times. i.e. an error occurring twice initiated the reflective prompt whereas an error occurring thrice triggered the conceptual prompt. Even though every attempt was taken to provide dialogue support according to the proposed structure, it was not always possible because students could respond freely in natural language and did not have to select from a list of possible answers as proposed in the model.

Participants interacted with ERM-Tutor in one room, while the interactions were observed from another room. The participants could initiate interventions through the chat interface or the *More Help* button. Participants were expected to use the system for at least an hour. However, students themselves decided when to end the session. At the end of the session, they filled out a questionnaire.

The first phase of the study involved analyzing the logs to investigate the effectiveness of the dialogues. In the second phase, the human tutors (acting as judges) were asked to judge the appropriateness of interventions by observing recorded sessions. A time line indicating all the interventions was provided to the judges, who indicated whether he/she agrees with the timing and the content of interventions. In the case of a disagreement, the judge was requested to provide justifications.

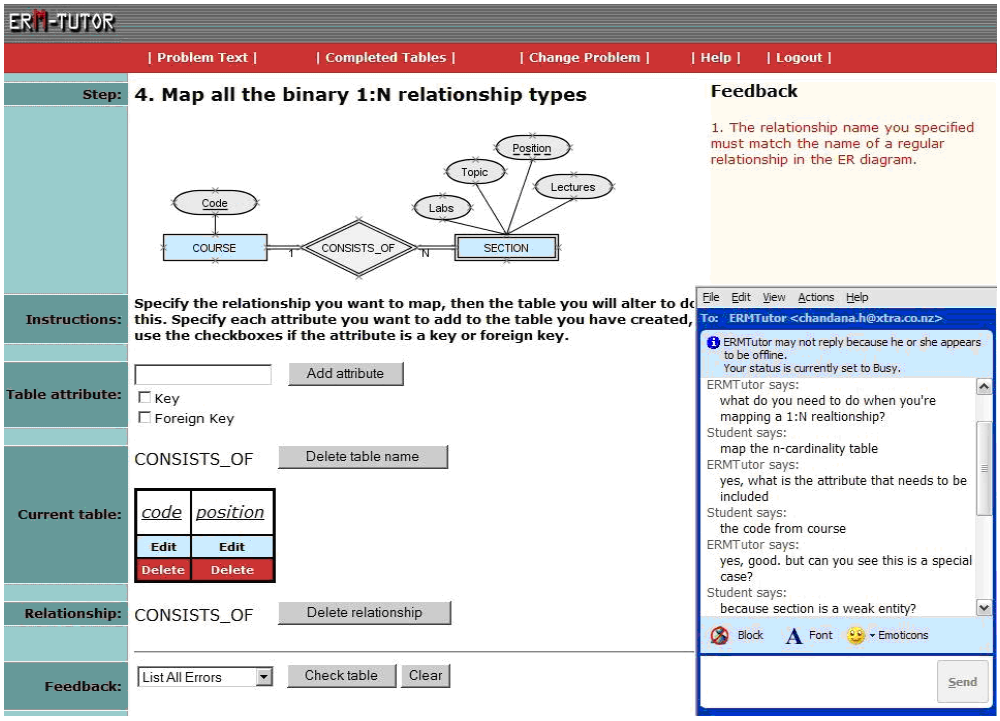


Figure 4.6: Interface of the enhanced ERM-Tutor used in the study

Results and Analysis – Phase 1: Ten students and five human tutors participated in the study. All students were enrolled in an introductory database course (COSC226) at University of Canterbury. The judges were the lecturer and the tutors involved in teaching this course.

In some cases, ERM-Tutor indicated that the student solution was incorrect even though it was actually correct, due to bugs in the system. Such instances were excluded from the analysis. The average session duration was 59 minutes (sd=15.3). The average number of problems attempted was 11 (sd =4.6), with 8.4 (sd = 5.2) problems completed on average.

From the logs, we identified 65 dialogues, each pertaining to a single topic, as in (Chi, 2000). In addition to facilitating remediation, some dialogues focused on helping with the interface (such as moving to the next step), completing the session or helping with technical problems (e.g. web browser not being able to display the page). The number of dialogues per session ranged from 1 to 13, with a mean of 6.5 (sd = 4.3). We are mainly interested in 31 dialogues that facilitated error remediation. Six of these dialogues contained a single utterance each, initiated by the wizard. For instance, a tutor utterance that helped a student to understand that multi-valued attributes are not mapped in the first step of the algorithm was "think about the colour attribute". The longest dialogue consisted of 11 utterances, 6 of which were provided by the model (i.e. simulated by the author).

An example is given in Figure 4.7. In this dialogue, the student is incorrectly applying step 4 (mapping 1:N relationships) to the identifying relationship, while this step should only be applied to regular relationship types. The correct action here is simply to move to the next step. In this situation, the model aims to assist the student to understand that this step is not necessary.

Tutor1:	What do you need to do when you're mapping a 1:N relationship?
Student1:	Map the n-cardinality table
Tutor2:	Yes, what is the attribute that needs to be included
Student2:	The code from course
Tutor3:	Yes, good. But can you see this is a special case?
Student3:	Because section is a weak entity?
Tutor4:	Yes

Figure 4.7: A dialogue from the study

Learning of constraints: In order to investigate whether the dialogues were effective in enhancing learning, we analysed how frequently an error occurred after being discussed in a dialogue. As the knowledge base in ERM-Tutor is represented as a set of constraints, the errors were recorded as violations of constraints. Thus we analyzed how frequently the constraints that were discussed in the dialogues were violated subsequently. However, some dialogues were excluded from the

analysis due to a coding problem. Therefore, the analysis only included remaining 15 (48.3%) dialogues.

The selected dialogues involved only seven participants. (The dialogues with the other three students were among the ones excluded.) These dialogues were associated with seven different domain-level constraints. Figure 4.8 illustrates the learning curve for these constraints. The curve is not smooth due to the small sample size (i.e. this analysis involved only seven constraints discussed in 15 dialogues with seven participants), but it does suggest that the frequency of subsequently violating a constraint discussed in a dialogue decreases with occasion number. This indicates that the students seem to learn domain concepts discussed in the dialogues, i.e. that the dialogues based on the proposed model did not have a detrimental effect on learning. In order to evaluate whether the dialogues facilitated by this model actually enhances learning we need to compare the performance with a control group of students who interact with the system without the dialogues.

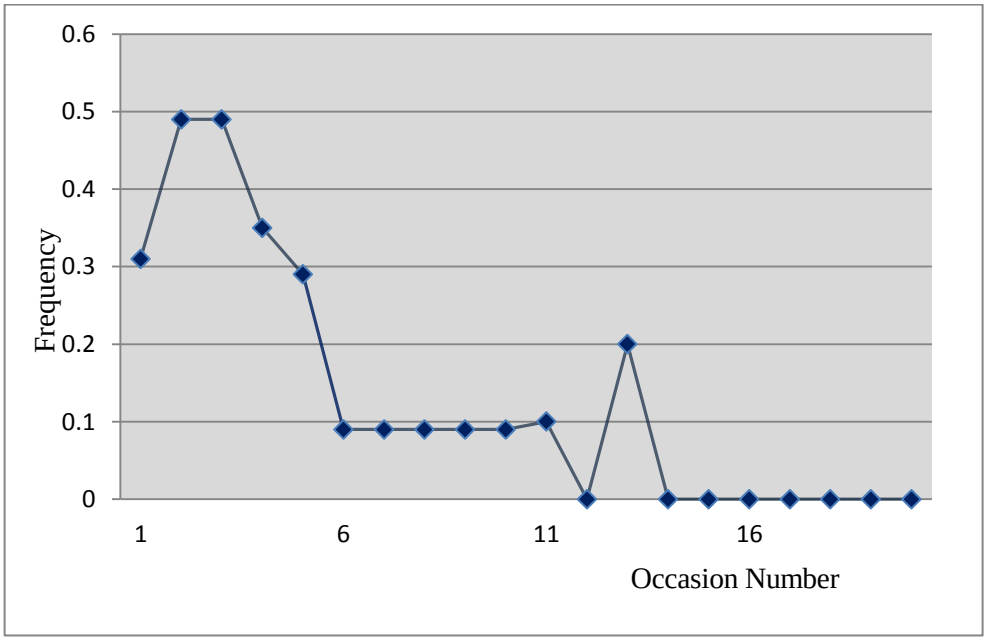


Figure 4.8: Learning from the dialogues in the Wizard-of-Oz study 2

Results and Analysis – Phase 2: Five judges analyzed the interventions and indicated whether they agreed with their timing and content in this phase. At the beginning, we informed the judges that the goal of the study was to develop a model to facilitate error remediation through dialogues

while interacting with a tutoring system. The judges were asked to comment on the appropriateness of the timing and the content of interventions provided through the chat interface. Each judge analysed two sessions. Due to time constraints, it was not possible for every session to be analysed by two judges.

All dialogues were categorized by the rule that initiated them. Five rules were relevant in this study. Rule 4 (triggered after a student has been inactive for 8 minutes) was triggered only once, and rule 5 (triggered after a student has been inactive for 4 minutes) was triggered three times. The judges agreed with the timing and content of these interventions.

Rule 2 (selecting the starting prompt of the dialogue) was relevant in 21 dialogues and had the highest number of disagreements. Judges disagreed in 7 (33.3%) occasions. Timing was the issue in six instances and judges wanted to intervene earlier. The judges disagreed with the content in three situations. For instance, a judge suggested using "*Is there a regular 1:N relationship to map in this problem*?" instead of the first prompt in Figure 4.7

The time period of a student's inactivity for which the model waits before intervening may have to be domain-dependant; however, there was no disagreement on this. Further investigation is needed before the model is changed.

Questionnaire responses: This section summarises the responses to the user questionnaire which participants completed at the end of study. The questionnaire had 16 questions, out of which eleven focused on the dialogue prompts. The remaining questions focused on enjoyment of the system, amount learnt etc. Similar to the questionnaire used in the Wizard-of-Oz study with EER-Tutor, we used a combination of close-ended, open-ended and hybrid questions. Three were strictly close-ended limiting the user responses to a set of pre-specified responses. Seven were open-ended proving an opportunity for participants to respond as needed. The remaining six were hybrid questions, consisting both close-ended (limiting responses to a Likert scale or pre-specified set of options) and open-ended parts. Table 4.1presents the mean responses.

The response to the question related to previous experience with the database mapping indicated that 80% (8 out of 10) of the volunteers have participated in lectures and engaged in some additional learning. Lectures have been the only source of learning for two participants. Even

though the previous experience indicated is approximately the same, the individual knowledge level can be different.

We now discuss the students' responses about dialogue prompts. The participants indicated (only five participants responded) that the dialogue prompts were helpful in identifying their mistakes themselves (mean response was 4.0 in Table 4.1). Participants who received dialogue prompts indicated that the prompts helped them to understand database mapping concepts (question 8(c) of the questionnaire). They also indicated that the prompts were easy to understand (mean response was 4.0 in Table 4.1). Therefore the user responses indicated that the dialogue prompts helped them to reflect on their own mistakes, learn domain concepts and were easy to understand.

Table 4.1: Mean responses from the questionnaire for Wizard-of-Oz study 2

	No of responses	**1 to 5 on Likert scale**	**mean**	**s.d.**
How helpful were dialogue prompts to identify your mistakes on your own	5	"Not at all" to "Very much"	4.0	0.71
How easy were the dialogue prompts to understand	4	"Not at all" to "Very easy"	4.0	0.71
How effective was the help through the chat interface compared to the typical ERM-Tutor feedback	10	"Help through the chat interface was very useful" to "Feedback by the ERM-Tutor was very useful"	2.2	1.14
Amount learnt (includes both help through the chat interface and the typical ERM-Tutor feedback)	10	"Nothing" to "Very much"	3.8	0.63
Enjoyment	10	"Not at all" to "Very much"	3.7	0.95

When asked about the timing of the interventions through the chat interface (question 6(a) of the questionnaire), 86% (7 out of 8 responded) of the participants indicated that the help was given at the right time. Only one participant indicated that the help was delayed more than necessary.

The responses to the question related to the level of detail provided by the dialogue prompts (question 6(b) of the questionnaire) indicated that help was given at the right level of detail. They appreciated that the prompts provided just enough detail to be guided towards the correct solution rather than the exact action to correct the mistake. However one participant indicated that he/she would have liked more detail on the next action to be performed. Furthermore, the participant who

felt that the feedback was delayed is not the same one who felt that more detail was needed on the next action to be performed.

When asked to compare the effectiveness of help through the chat interface and the typical feedback of ERM-Tutor, the mean response was 2.20. Two participants indicated that the typical feedback by the ERM-Tutor was good for getting immediate feedback for basic problems. This response might be due to the slight delay in getting feedback initiated by the model (i.e. time required for typing the response). The same participants felt that the feedback initiated by the model provided insightful information and was better for harder problems. This might be due to the type of questions the model initiated such as the reflective prompts etc.

Two questions addressed different aspects such as the enjoyment and amount learnt. When asked how much they learnt about logical database design using ERM-Tutor (which includes both the typical feedback by the system and the help thorough the chat interface), the mean response was 3.8. They also indicated that they enjoyed learning with ERM-Tutor with a mean response of 3.7. Furthermore all the participants indicated that they would recommend ERM-Tutor to others (question 4 of the questionnaire)

Discussion: The goal of this study was to try out the model in the domain of logical database design prior to its full implementation. Analysis of user logs indicates that students did learn the domain concepts discussed in the dialogues. Human tutors who were asked to analyze the dialogues mostly agreed with the interventions generated by the model.

We made further improvements to the dialogues based on the findings of the study. For example, one of the improvements focused on how to effectively facilitate error remediation when nothing needs to be done in a particular step. According to our model, the initial prompt when mapping a 1:N relationship (step 4 of the mapping algorithm) was "What do you need to do when you're mapping a 1:N relationship?" which may imply that the student needs to perform an action, even if none is needed. The prompt would be clearer and would not imply any required action if it was changed to "*Do you know which type of relationship needs to be mapped in this step*?". The new prompt discusses a domain concept so it still conforms to the proposed dialogue structure (Section 3.3.2).

Even though our Wizard-of-Oz studies investigated providing dialogue support for two instructional tasks (ill –defined task of conceptual database design and well-defined task of logical database design), mapping is the main activity in both these tasks; i.e. in conceptual database design students are expected to map a problem statement given in natural language to a data model. Logical database design requires mapping from a data model to a database schema supported by the chosen DBMS. Therefore, in order to explore the applicability of our model in a different type of task, data normalization was chosen. This is a well-defined task, which is very different to both conceptual database design and logical database design. We enhanced NORMIT, with tutorial dialogues using our model. The evaluation study conducted to evaluate the effectiveness of dialoged-enhanced NORMIT are presented in Section 4.2.2

4.2 Full Scale Evaluations

In order to evaluate the generality of our model in providing tutorial dialogue support, we implemented the model in EER-Tutor and NORMIT, two existing constraint-based tutors that teach two different types of instructional tasks. EER-Tutor teaches conceptual database design, an ill-defined task with a well-defined domain theory (a domain in the WDIT quadrant as discussed in Section 3.1.1). On the other hand, data normalization is a well-defined task with a well-defined domain theory (a domain in the WDWT quadrant). Thus by implementing the model in both an ill- and a well-defined task, we would be able to evaluate the model's ability to support dialogues in multiple domains. In order to evaluate the effectiveness of the adaptation of dialogues, we compared the learning of a group of students who received adaptive dialogues with their peers who received non-adaptive dialogues. In the first study both groups interacted with EER-Tutor. The error hierarchy and the dialogues used to discuss errors were the same for both groups. The only difference between the two groups was adaption of dialogues. Thus the difference in the learning between the two groups can be attributed to the adaptation of the dialogues. We repeated the experimental design with NORMIT to investigate the effect of adaptive dialogues on learning data normalization. This type of evaluation corresponds to evaluating all the layers of an IAS, the last adaptation layer of the framework for evaluating IASs (Section 3.2). Suitable methods of evaluation are focus group, cognitive walkthrough, heuristic evaluation and user test. When the

implementation of the functionality corresponding to an evaluation layer is complete, user tests can be used. Thus we chose user tests. They enable us to understand how students interact with the ITS. The remainder of this Chapter presents the details of these studies.

4.2.1 EER-Tutor Study

We conducted a study with EER-Tutor at the University of Canterbury[2], which involved volunteers from an introductory database course. The objective of the study was to investigate whether adaptive dialogues are more effective in improving learning conceptual database design than non-adaptive dialogues. The participants used EER-Tutor for the first time in their regular lab sessions during the third week of the course, by which time they had been introduced to EER modelling.

Experimental design: The participants were randomly assigned to two groups (experimental and control). The experimental group received adaptive support based on our model. The control group was given non-adaptive support in which two different students with different knowledge levels received the same dialogue for identical attempts. Differences between the two groups were: (i) Dialogue selection, (ii) Dialogue prompts and (iii) Other support provided by the model.

Dialogue selection: The dialogue selection for the control group was based on finding the simplest error for discussion. As the errors in the hierarchy were ordered from simpler to more complicated errors, the first violated constraint that was found corresponds to the simplest error in a student solution (Section 2.4). For instance, consider the student solution in Figure 2.7(a) for the problem statement in Figure 2.6. The error selected for discussion was that CHAPTER was modelled as a regular entity. The corresponding dialogue is presented in Figure 4.9(a).

Now consider an experimental group participant with an identical student model to the previous student submitting the same solution. According to the model, incorrect cardinality between the entity *TEXT_BOOK* and the relationship *CONTAINS* was chosen as the

[2] This study was approved by the Human Ethics Committee of the University of Canterbury. The approval letter is given in Appendix C.1.

pedagogically most suitable error to focus upon. Figure 4.9(b) presents the corresponding dialogue.

Dialogue prompts: The control group saw the entire dialogue regardless of the number of times they have seen the dialogue previously in the session or their responses to the dialogue prompts. As a result, the same solution submitted by two different students with different knowledge levels in the control group received the same dialogue. Consider the example mentioned above: the student solution in Figure 2.7(a) for the problem statement in Figure 2.6. The dialogue started from conceptual prompt (EER-Tutor1 in Figure 4.9 (a)) for the control group as they received the entire dialogue every time the same mistake is repeated. In contrast, the dialogue started from the reflective prompt (EER-Tutor2 in Figure 4.9 (b)) for the experimental group as it was the first time the error was made during the current session. If the same error was made 3 times or more within the current session, the dialogue starts from the conceptual prompt (EER-Tutor1 in Figure 4.9(b)).

Other support provided by the model: When an experimental group participant abandons a problem (i.e. changes a problem without submitting at least once) or has been inactive for a period of time, the student solution was evaluated. The student was also asked whether he/she needed help. If help is requested, pedagogically most suitable error for discussion was selected and the corresponding dialogue was initiated. The control group did not receive this support.

Stages of the study: The study consisted of four stages: (i) pre-test (Appendix C.2), (ii) interactions with EER-Tutor, (iii) post-test (Appendix C.3) and (iv) questionnaire (Appendix C.4). At the beginning of the study, the participants were randomly allocated to the control and experimental groups. The duration of the study was limited to two hours. At the beginning of the study students were given about 10 minutes to complete the pre-test. Then they started solving problems using EER-Tutor. Towards the end of the session, they were given 10 minutes to complete the post-test and 5 minutes to answer a questionnaire.

Pre- and post-tests: Pre-tests were used to determine the participants' knowledge before interacting with the system and also to determine whether the knowledge between the experimental

and control was significantly different. Both pre- and post-tests had six questions each. The questions in the pre- and post-tests were of similar difficulty. We wanted to evaluate whether students' problem-solving abilities as well as explanation skills improved after interacting with the system. One question asked the participants to provide the database schema for the given requirements. This is a typical question that can be found in examinations, text books etc. Three questions were aimed to understand the effect the system had on students' explanation skills. The remaining two questions asked about declarative knowledge.

Figure 4.9 A hypothetical dialogue for EER-Tutor received by a (a) control group participant (b) experimental group participant

(a)	(b)
EER-Tutor1: You seem to be having some difficulty with regular entities. Can you tell me the general rule to decide whether something is a regular entity?	**EERTutor1:** Looks like you have trouble with participation. Let's focus on participation now. What do you represent by total participation on the E1 side between entities E1 and E2 in a binary relationship?
Student1: *I'm not sure*	**Student1:** I'm not sure
EER-Tutor2: Let's try a simple thing. What is used to uniquely identify instances of a regular entity?	**EERTutor2:** Here's a simple question to try. How do we model connections between entities?
Student2: *key attribute*	**Student2:** Using relationships
EER-Tutor3: Great! Now tell me why modelling *CHAPTER* as a regular entity is incorrect?	**EERTutor3:** Great! Why is the participation between *TEXT_BOOK* and *CONTAINS* is wrong?
Student3: *I'm not sure*	**Student3:** I'm not sure
EER-Tutor4: Let's see …Check carefully whether you can find a key attribute for *CHAPTER*. {After *Continue* button is pressed} Please select the key attribute for *CHAPTER*?	**EERTutor4:** Let's see…. Try to figure out whether every instance of *TEXT_BOOK* participates with instances of *CHAPTER*") {After Continue button is pressed} Now tell me how many instances of TEXT_BOOK participates with instances of CHAPTER"
Student4: *No key attribute for CHAPTER*	**Student4:** Exactly one
EER-Tutor5: Good job! So now can you tell me the best way to model *CHAPTER*?	**EERTutor5:** Well done! Guess you know how to make change now. Before that, try to answer this question. What is the correct question to ask when deciding the participation on the E1 side between entities E1 and E2 in a binary relationship?
Student5: weak entity	**Student5:** Does every instance of E1 participate with instances of E2?
EER-Tutor6: Well done! Now use the correct word to fill in the blanks so that you'll remember what you learnt. A …….. uniquely identify instances of a regular entity…….. in an ER model	**EERTutor6:** Great job! go ahead and do the changes. Submit the solution when you're done.
Student6: key *attribute,* regular entity	
EER-Tutor7: Great job! Please make the changes and submit your solution when you're done.	

Questionnaire: There were eight questions aimed at understating participants' views of different aspects of dialogues. Five questions had Likert scales (ranging from 1 to 5) discussing the quality, the length and the prompts in the dialogues. Participants were also given an opportunity to explain how the dialogues helped them in their learning. Some questions focused on suggestions to improve the dialogues and EER-Tutor in general.

Results and Analysis: Out of 104 students enrolled in the course, 77 participated in the study. Only 65 participants (31 participants in the experimental group and 34 in the control group) completed both pre- and post-tests. Table 4.2 reports some statistics about the 65 participants.

Table 4.2. Some statistics from the EER-Tutor study (sd given in parentheses)

	Control group (34)	Experimental group (31)	Level of significance p
Pre-test (%)	54.5 (18.1)	51.3 (16.1)	ns
Post-test mean (%)	61.2 (14.9)	69.9 (11.5)	0.005
Gain	6.8 (15.6)	18.6 (16.8)	0.002
Normalised gain	0.002 (0.7)	0.3 (0.4)	0.01
Interaction time (min)	62.8 (22.1)	62.9 (24.1)	ns
Attempted Problems	8.6 (4.8)	10.6 (4.8)	ns
Solved problems	9.0 (4.8)	7.9 (4.7)	ns
Total Dialogues received	12.1 (7.3)	14.0 (8.3)	ns
Single-level dialogues seen	2.1 (3.0)	1.9 (2.7)	ns
Multi-level dialogues seen	10 (6.8)	12.1 (7.2)	ns
Total number of prompts answered	34.4 (25)	23.6 (14.6)	0.01
Number of prompts answered correctly	23.3 (17.9)	14 (10.4)	ns
Prompts answered correctly (%)	61.4 (23.1)	59 (16.9)	ns
Number of prompts answered incorrectly	9.1 (8.3)	7.3 (4.3)	ns
Prompts answered incorrectly (%)	23.7 (12.9)	31.8 (15)	0.01
Number of prompts answered with a *More Help* request	2.1 (3.5)	2.4 (3.5)	ns
Prompts answered with a *More Help* request (%)	6.1 (6.9)	9.22 (11.4)	ns

The two groups were comparable as there was no significant difference between the pre-knowledge of the two groups. The post-test performance of the students who received adaptive dialogues increased significantly more than their peers who received non-adaptive dialogues (t =

2.6384, p = 0.005). Both the learning gain[3] (t= 2.9474, p = 0.002) and the normalised learning gain[4] (t = 2.1511, p = 0.01) of the experimental group are also significantly higher than the other group.

The two groups spent a similar amount of time interacting with the system. This is consistent with the experimental design as the study was limited to a single lab session. There is also no significant difference between the number of attempted and solved problems. The total number of dialogues, the total number of single-level dialogues (some dialogues are limited to a single feedback message as they discuss simple errors) and the total number of multi-level dialogues are also similar for the two groups.

The control group answered a significantly higher number of prompts than their peers (t = -2.1571, p = 0.01). This was expected, as the control group had to go through the entire dialogue before resuming problem solving. However, percentages of correct answers are similar for the two groups. There are no significant differences on the total number of prompts answered incorrectly or the prompts with a "More Help" request (i.e. one of the options available was "I don't know" or "I need more help" which resulted in presenting the relevant information to the student). Also there was no difference on the percentage of prompts that requested more help. However, it is interesting to note that the experimental group has provided a significantly higher percentage of incorrect answers (t = 2.3304, p = 0.01). This is to be expected as the dialogue selection for the experimental group was based on the error type that was most frequently made for each individual student. On the other hand, dialogues given to the control group participants were based on the simplest error on the pre-specified error hierarchy. As a result, experimental group had to answer prompts that were difficult for them, whereas the prompts answered by their peers in the control group may or may not be difficult.

The effect size is a standard way to compare the results of one pedagogical experiment to another. The common method to calculate the effect size in the ITS community is to subtract the control group's mean gain score from the experimental group's mean gain score and divide by the standard deviation of the gain scores of the control group (Bloom, 1984). This calculation results in an effect size of 0.69 (the effect size based on the normalized gain is 0.51). This is comparable to

[3] Learning gain = post-test score – pre-test score

[4] Normalised learning gain =learning gain/(1-pre-test score)

the study with SQL-Tutor conducted in a similar setting in a single 2-hour session (Mitrovic, Martin, & Mayo, 2002). An effect size of 0.66 was reported for that study for the students who used SQL-Tutor compared with those who did not use the tutor. The effect size obtained here is therefore remarkable because the only difference between the two groups was the adaptivity of the dialogues.

Learning Curves: In order to investigate how the students in both groups learnt the database design concepts in terms of constraints, we analyzed how frequently constraints were violated. Figure 4.10 shows the learning curves for both groups. The frequencies of violating constraints on the first and subsequent occasions were averaged over all students. The X-axis represents the occasion number (first, second and so on) when a constraint is relevant. The Y-axis shows the frequency of violating these constraints. Figure 4.10 indicates that both groups learnt the constraints in a similar manner. Both learning curves have a good fit to the power curve, indicating that the transferability of learning is high for both groups.

Another measure of learning is the number of newly learnt constraints during the session. The number of newly learnt constraints is determined heuristically. The heuristic only considers the first 5 occasions and the last 5 occasions during which a constraint is relevant. The frequency of violating a constraint is calculated separately for the first 5 and the last 5 occasions. If the frequency of violating a constraint is below a pre-defined threshold then the constraint is considered to be known, otherwise not known. A constraint is said to be newly learnt if it is not known at the beginning of the session (based on the first 5 occasions) and known at the end (based on the last 5 occasions). This analysis revealed that the experimental group learnt a significantly higher number of constraints than the control group (2.3 vs 1.2 $t = 2.2063$, $p = 0.02$).

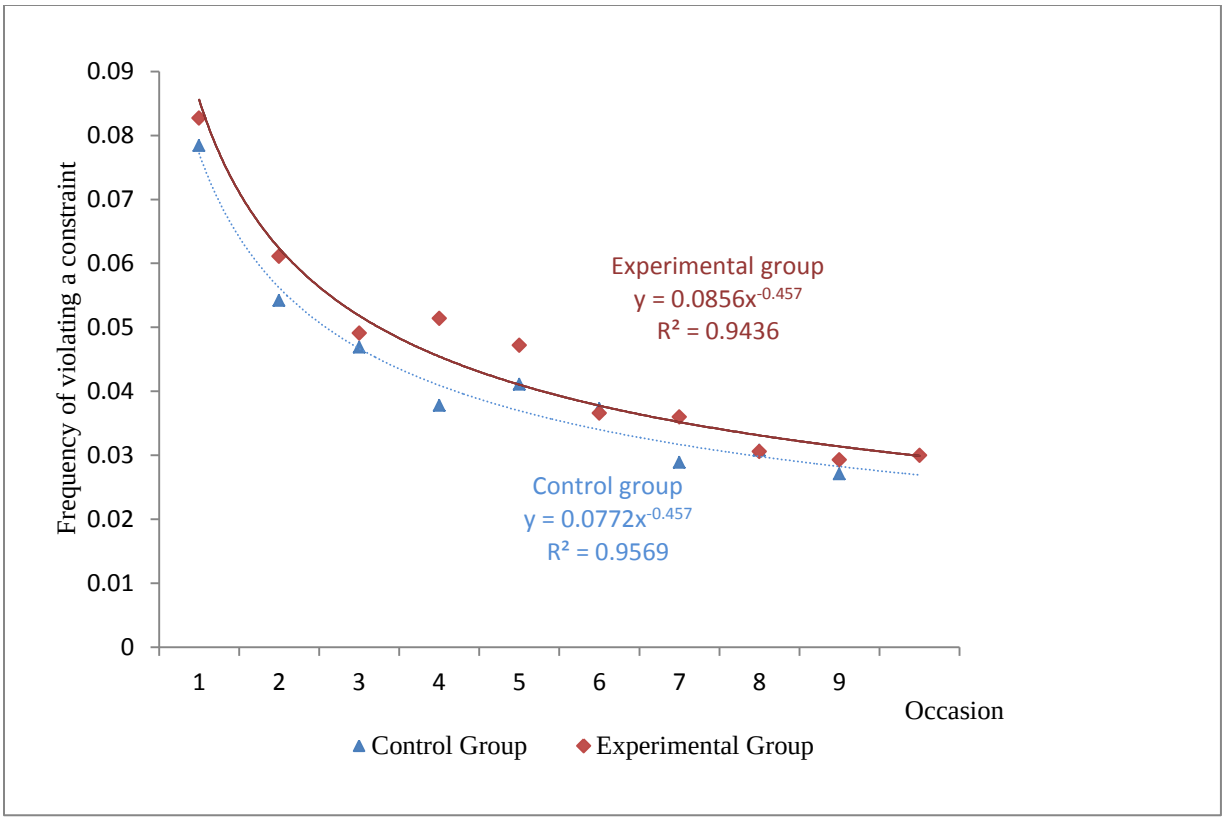


Figure 4.10: Frequency of constraint violations - EER-Tutor study

Subjective Responses: Table 4.3 presents the subjective responses about various aspects of the dialogues. The analysis based on the Mann-Whitney U test on the questionnaire responses revealed that the impressions about the quality of the dialogues and the ease of understanding the prompts were similar between the groups. However there was clear evidence that the control group did not like having to go through the entire dialogue (Mann-Whitney U test, U = 299, p = 0.009).

Table 4.3: Subjective responses about tutorial dialogues for EER-Tutor study

Question	Likert scale (1 to 5)	Control group (34)	Experimental group (31)	Level of significance p
Overall quality of the dialogues	Poor to Excellent	3.5 (1.0)	3.7(0.8)	ns
Length of the dialogues	Too long to Too short	2.6(0.9)	3.2(0.5)	0.009
Ease of understanding the prompts	Very hard to Very easy	3.1(1.0)	3.4(0.8)	ns

4.2.2 NORMIT Study

We conducted a study with NORMIT in September 2010 at the Victoria University of Wellington[5], which involved volunteers from a database system engineering course.

Experimental Design: The objective and the experimental setup for this study are similar to that of EER-Tutor study. Both pre- and post-tests had 4 questions each. The questions in the pre- and post-tests were of similar difficulty. Similar to the previous study, we wanted to evaluate whether students' problem-solving abilities as well as explanation skills improved after interacting with the system. Two questions asked the students to determine the candidate keys for the given schema and to justify their answers. The other two questions asked about declarative knowledge.

Results and Analysis: 20 students participated in the study. Two students did not complete the post-test. Table 4.4 reports statistics about the 18 participants who completed both pre-and post-tests. Experimental and control groups had nine students each. The lab session was scheduled for an hour.

There were no significant differences between the pre-test and post-test performances of the two groups, as well as between the gains. The performance of the experimental group increased significantly from pre-to post-test (paired t-test, $t=1.84$, $p=0.052$), while the improvement of the control group was not significant. The effect size for learning gains of the two groups is 0.4.

The two groups spent a similar amount of time interacting with the system, consistent with the study being limited to a single lab session. Both groups attempted and solved a similar number of problems and received a similar number of single-level and multi-level dialogues.

The control group participants answered significantly more prompts than their peers ($t = -2.4357$, $p = 0.01$), as was the case in the EER-Tutor study. This can be expected as the control group had to go through the entire dialogue every time a dialogue is initiated. However, the percentage of correct answers and incorrect answers were similar for both groups. There is

[5] This study was approved by the Human Ethics Committee of the Victoria University of Wellington. The approval letter is given in Appendix D.1

no difference on the percentage number of prompts with a "More Help" request which resulted in providing the relevant information to the student.

Table 4.4: Some statistics from the NORMIT study (sd given in parentheses)

	Control group (9)	Experimental group (9)	Level of significance p
Pre-test (%)	68.1 (30.0)	69.4 (29.4)	ns
Post-test mean (%)	72.2 (24.0)	86.1(15.9)	ns
Gain	4.2 (32.4)	16.7 (27.2)	ns
Interaction time (min)	60.1(24.7)	47.7 (16.8)	ns
Attempted Problems	7.1 (3.0)	5.9 (2.1)	ns
Solved problems	6.1 (3.0)	5.4 (2.0)	ns
Total Dialogues received	27.8 (14.6)	23.6 (11.3)	ns
Single-level dialogues seen	12.3 (7.4)	13 (10.6)	ns
Multi-level dialogues seen	15.4 (9.9)	10.5 (4.7)	ns
Total number of prompts answered	55.7 (37.4)	23.9 (11.5)	0.01
Total number of prompts answered correctly	34.8 (20.4)	17.6 (10.1)	0.02
Prompts answered correctly (%)	6.9 (4.1)	8.2 (4.7)	ns
Total number of prompts answered incorrectly	14.6 (14.1)	4. 6 (3.3)	0.03
Prompts answered incorrectly (%)	2.9 (2.8)	2.1 (1.5)	ns
Total number of prompts with a *More Help* request	6.3 (9.0)	1.8 (1.2)	ns
Prompts answered with a *More Help* request (%)	1.3 (1.8)	0.8 (0.6)	ns

Learning Curves: Figure 4.11 represents the learning curves for both groups which indicate how the participants learnt the data normalization in terms of constraint violations. The frequency of making a mistake is initially higher for the experimental group than the control group even though not significant. The learning curves indicate that both groups learnt the constraints in a similar manner. However, the experimental group learnt at a higher learning rate (-0.116 for the control group vs -0.34 for the experimental group).

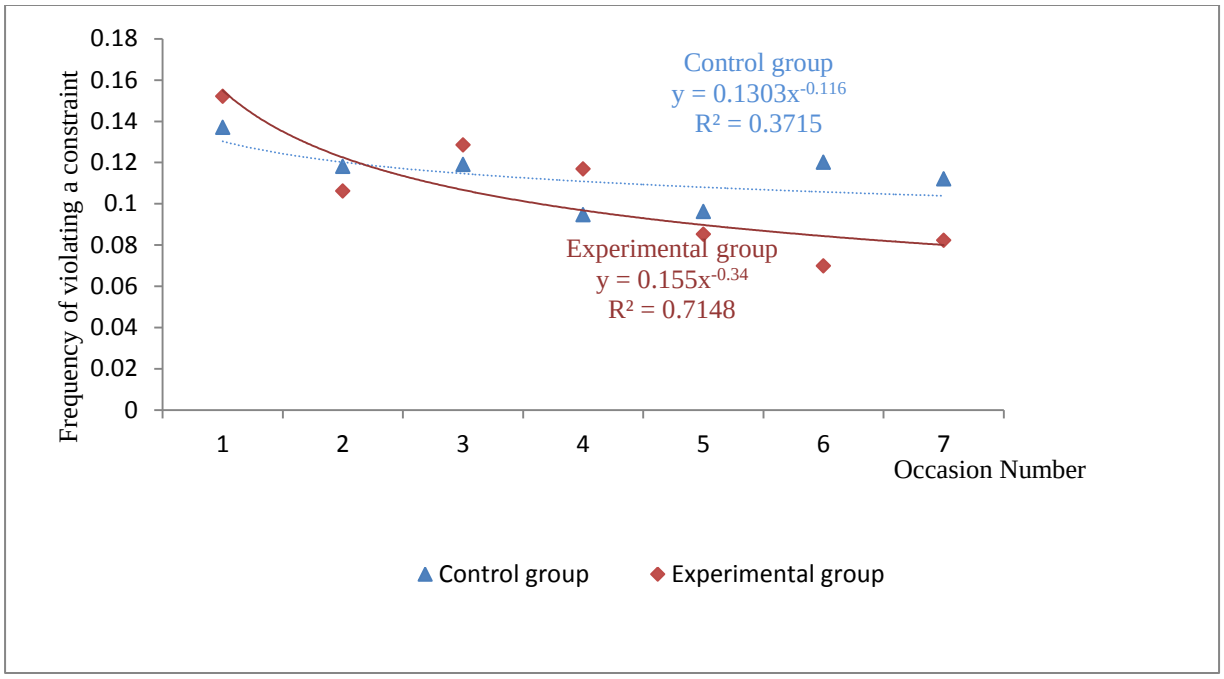


Figure 4.11: Frequency of constraint violations - NORMIT study

Subjective Responses: Table 4.5 presents the subjective responses about various aspects of the dialogues. Similar to the responses of the EER-Tutor study, impression about the quality of the dialogues and the ease of understanding the prompts were similar between the groups. In contrast to the EER-Tutor study, the control group did not indicate that the length of the dialogues was too long. However, this result may be due to the small sample size.

Table 4.5: Subjective responses about tutorial dialogues for NORMIT-study

Question	Likert scale (1 to 5)	Control group (9)	Experimental group (9)	Level of significance p
Overall quality of the dialogues	Poor to Excellent	3.3(0.5)	3.1(1.0)	ns
Length of the dialogues	Too long to Too short	3.1 (0.8)	3.3(0.5)	ns
Ease of understanding questions	Very hard to Very easy	3.4 (0.7)	3.1(0.7)	ns

4.3 Discussion

We used paper-based investigations in four domains (conceptual database design, data normalization, logical database design and fraction addition) to evaluate the generality of the first

two components of the model (i.e. the error hierarchy and the tutorial dialogues). These investigations involved two tasks: (i) developing an error hierarchy to categorise all errors in the domain and (ii) developing dialogues to discuss the errors specified in each of the error hierarchies. We provided evidence that (i) the error hierarchy for each domain categorises all error types for that domain and (ii) proposed dialogue structure is expressive enough to discuss all the errors in each of the domains. Each domain that we focused on is associated with a well-defined domain theory. However, conceptual database design is the only ill-defined task, all the others are well-defined. Thus we have provided evidence of the generality of the first two components of our model in four domains.

The final component of the model, adaptation rules, was developed based on the students' interactions with EER-Tutor learning conceptual database design. Prior to the full implementation of the model we evaluated the effectiveness of the prototype of the model using ERM-Tutor. The first Wizard-of-Oz study involved teaching conceptual database design, an ill-defined task with a well-defined domain theory, whereas second study focused on logical database design, a well-defined task with a well-defined domain theory. Thus we provided evidence of the generality of the adaptation rules in two different domains involving two different types of tasks.

Then full scale evaluations were used to evaluate the effectiveness of our model for supporting tutorial dialogues in two different types of tasks. In the EER-Tutor study, the learning gain of the experimental group (who received adaptive dialogues) was significantly higher than the gain of their peers, with the effect size of 0.69. The analysis of the number of constraints learnt during the session (discussed in Section 4.2.1) revealed that the experimental group learnt a significantly higher number of constraints than their peers in the control group. These results strongly suggest that adaptive dialogues had a positive effect on learning conceptual database design. This is a significant result because (i) the difference between the two groups was small (i.e. the only difference was the adaptivity of the dialogues) and (ii) the study was limited to 2-hours.

In the NORMIT study, there were no significant differences between the pre-test and post-test performances of the two groups, as well as between the gains. This might be due to the small number of participants (18 vs. 65 in EER-Tutor study). However, we observed a similar trend in

learning: a higher learning rate in NORMIT study by the respective experimental group compared to their peers.

Dialogues were used to discuss errors in the student solutions for both groups (experimental and control) in both studies. i.e. the tutoring systems took advantage of good learning opportunities as the previous research suggest that student learn when they reach an impasse (Ohlsson, 1996; VanLehn et al., 1998). The only difference between the two groups was the adaptation rules that drives dialogue selection and its content. We now discuss why the adaptation rules had a positive effect on learning of the experimental group.

Selecting the pedagogically most suitable error to discuss was based on the constraint histories for the experimental group whereas it was based on the current attempt for the control group. Therefore stronger evidence was used to select of the most suitable error for the experimental group. In addition, error selection was based on the error type that was most frequently made for each student in the experimental group whereas it was simplest error for the control group. Thus the error type that was most frequently made points to a gap or a misconception in the student knowledge. On the other hand, the simplest error based on current attempt may or may not point to a knowledge deficit; it can be a slip. As the result, the dialogues received by the experimental group had potentially fulfilled a gap or a misconception in their knowledge. Even though the adaptation rules required the experimental group to focus on the error type that was most frequently made, their performance increased significantly than the control group. This result agrees with previous research that claims making performance more difficult during instruction improves learning (Koedinger & Aleven, 2007).

Furthermore, the experimental group had the opportunity to acquire the domain knowledge faster because the chosen dialogues focused on the error type that was most frequently made. This was supported by the significantly higher number of constraints learnt by the experimental group in the EER-Tutor study.

When an error was made for the first time, the starting prompt received by the experimental group was reflective, whereas it was conceptual for the control group. The experimental group was required to use both procedural and conceptual knowledge to answer the reflective prompt correctly. In addition, the prompt discusses the selected error within the current problem-solving context. In contrast, the control group was asked to focus only on the conceptual knowledge

corresponding to the selected error and the prompt was not related to the current context. As the result, the control group participants either had to relate the knowledge gained from the conceptual prompt to the current context themselves.

Each dialogue provided different opportunities to learn: the corresponding domain knowledge, how to correct the error or reason about their problem solving. However all of these different learning opportunities were presented one after another to the control group and some participants might have been overwhelmed and perceived the dialogue as a lengthy intervention which hinders them from focusing on problem solving. In contrast, the experimental group received the dialogue in increasing level of detail and they received the entire dialogue only after three repeated mistakes of a single error within a single session.

In both studies we used dialogues to discuss the errors in the problem-solving process, and not as the main activity to learn the domain knowledge. The task facilitated in EER-Tutor requires knowledge about different real-world scenarios such as enrolling students in a university, or customers interacting with a bank. In the EER-Tutor study, the model was used to support dialogues in an ill-defined task with a well-defined domain theory. In the NORMIT study, dialogues facilitated learning a well-defined task with a well-defined domain theory. Therefore, our model has shown evidence of enhancing learning of a domain in the WDIT quadrant and WDWT quadrant. Furthermore we have provided evidence about the model's ability to support dialogues in multiple domains by implementing the model in both an ill- and a well-defined task.

Chapter 5

Conclusions

Tutorial dialogues are highly interactive, which is a key characteristic in one-on-one human tutoring. Dialogues also provide opportunities to reflect on existing knowledge as well as to integrate new knowledge (Chi et al., 2001). Several ITSs have utilised dialogues to teach conceptual knowledge during problem solving (Aleven & Koedinger, 2002; Aleven et al., 2003; Mitrovic, 2005; Evans & Michael, 2006; Heffernan et al., 2008; Weerasinghe & Mitrovic, 2006; VanLehn et al., 2010). However none of the approaches used for generating dialogues have been used in multiple domains. This research explored the feasibility of developing a general model to facilitate adaptive dialogue support across domains. This model emulates the behaviour of human tutors by providing adaptive dialogues during problem solving in response to errors. The other focus of our research is to investigate whether adaptive dialogues are better than non-adaptive dialogues.

During our exploration we have made several contributions to ITS research. The main contribution of this research is a model that facilitates adaptive tutorial dialogues in multiple domains. We summarise the main contributions in Section 5.1. The same section includes a summary of the evaluations that produced very promising results. Limitations of this research are discussed in Section 5.2, followed by future research directions that build on the outcomes. Finally some closing remarks are given in Section 5.4.

5.1 Main Contributions

Our model provides adaptive dialogue support by identifying concepts that the student had most difficulty with, based on the student model, and then selecting tutorial dialogues corresponding to those concepts. Additionally dialogues are customised based on the student's knowledge and explanation skills, in terms of the length and the exact content of the dialogue. In contrast to existing ITSs that customise dialogues based only on tutoring history, our model uses a novel

approach that combines both the history of the tutoring session with the student model. The model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level an error type is associated with one or more violated constraints, which form leaves of the hierarchy. Error types are then grouped into higher level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When a student solution has multiple errors, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue, learners are able to provide answers by selecting the correct option from a list.

The three highest levels of the error hierarchy (the first component of the model) are domain-independent. Further divisions of these nodes are associated with domain-specific concepts. Even though dialogues consist of domain-specific prompts, their structure is domain-independent. Adaptation rules (the last component) which customise dialogues are domain-independent except for the time period of student inactivity the ITS waits before intervening.

After developing the model, the next step was to provide evidence of our model's capability for facilitating tutorial dialogues in different types of instructional domains and tasks. The evaluations involve four domains of different complexities: conceptual database design, logical database design, data normalization and fraction addition. Conceptual database design involves developing a schema for the database that satisfies a given real-world scenario. Even though database design is based on a well-defined domain theory, the task of database design is considered ill-defined because the final outcome is defined only in abstract terms without an algorithm to achieve the outcome. Data normalization uses a deterministic algorithm to refine a relational database to ensure that all relations are of high quality. Thus data normalization is a well-defined task based on a well-defined domain theory. Logical database design, the third domain used in the evaluations, involves mapping a high-level, conceptual ER schema to a relational schema using the 7-step mapping algorithm. Thus logical database design is a well-defined task associated with a well-defined domain theory. The final domain is fraction addition, which is a simpler task compared to the ones related to database modelling mentioned above, yet it is very different. In summary, each of these domains has a well-defined domain theory. However, all the instructional

tasks except conceptual database design are well-defined due to the well-defined procedures that should be used to achieve the final outcomes.

Two types of evaluations were carried out: paper-based investigations and full-scale evaluations in authentic classroom environments. Paper-based investigations involve two tasks. The first task is to explore whether an error hierarchy can be developed using an existing constraint base for the domain. Development of the error hierarchy starts with the top-three levels that are domain-independent. The second task is to explore whether all the errors in the domain can be discussed using the proposed dialogue structure. The proposed dialogue structure consists of four types of prompts: conceptual prompt, reflective prompt, corrective-action prompt and conceptual reinforcement prompt. The first paper-based investigation involved EER-Tutor, a constraint-based tutor that teaches conceptual database design. The second investigation used the constraint base of NORMIT, a constraint-based tutor that teaches data normalization, whereas the third involved ERM-Tutor that teaches logical database design. The final investigation used the constraint base designed to teach fraction addition. This set of constraints were developed by ASPIRE, an authoring tool to develop constraint-based ITSs. The result of each investigation was an error hierarchy that categorizes all the error types in the domain and a set of dialogues that discusses all the error types in the domain using the proposed dialogue structure. Thus we showed that the proposed domain-independent parts of the error hierarchy is applicable for the four domains and enables the development of an error hierarchy that categorises all the error types in each domain. In addition we also showed that the proposed dialogue structure is powerful enough to discuss all the error types in each domain.

Full scale evaluations involve incorporating our model into two existing constraint-based tutors and evaluating the effectiveness of the model in a real classroom environment with authentic students. Firstly, we incorporated our model into EER-Tutor and the effectiveness of the resultant system, dialogue-based EER-Tutor was evaluated in the first study. This study involved undergraduate students learning conceptual database design using the dialogue-based EER-Tutor. The results revealed that the acquisition of the domain knowledge (represented as constraints) of the experimental group who received adaptive dialogues was significantly higher than their peers in the control group with non-adaptive dialogues when both groups spent a similar amount of time with the system. The improvement of the experimental group in terms of the problem-solving

performance was significantly higher than the control group. Secondly, we incorporated our model into NORMIT and repeated the experiment in the context of dialogue-based NORMIT with a much smaller group of students (18 in NORMIT study vs 65 in EER-Tutor study). Even though the results indicated that the rate of learning of the experimental group was higher than that of the control group, the difference was not significant. Both groups also learnt a similar number of constraints.

Our attempts both in terms of evaluation studies and investigations on paper indicated that our model can provide adaptive support for both ill- and well-defined tasks associated with a well-defined domain theory. The results also indicate that adaptive dialogues are more effective than non-adaptive dialogues in teaching the ill-defined task of database design.

We now discuss other contributions of this research in addition to the model facilitating tutorial dialogues. We discussed why it is important to differentiate between a task and a domain when exploring the ill-definedness/well-definedness of a task and a domain. We also proposed how the space of instructional domains and tasks could be divided into four quadrants.

We built a hierarchy that categorizes all errors of a domain. We provided evidence that this hierarchy could be used in different domains: conceptual database design, logical database design, data normalization and fraction addition. Each of these domains is associated with a well-defined domain theory, but teaches different types of instructional tasks. i.e. conceptual database design is an ill-defined task whereas all the others are well-defined. This hierarchy is unique because it does not require exploring misconceptions in a domain: it categorises errors as instances where domain principles are not being adhered to. It is based on the assumption that a correct solution cannot be achieved by traversing a state that violates any of the domain principles.

Selection of the pedagogically most suitable error for discussion is based on a student's demonstrated ability for each domain concept. Error selection to provide feedback in existing dialogue-based systems are based on either a student's last attempt (VanLehn et al., 2000; Evans & Michael, 2006) or his/her overall ability of the task (Person et al., 2001). In contrast, our model keeps track of a student's ability to apply each domain concept in the problem-solving process since he/she starts using the constraint-based tutor, but considers only the more relevant short-term view for error selection (i.e. a student's ability to apply a domain concept is based on the last five opportunities to apply that constraint). Thus selection of the most suitable error is based not only

on a fine-grained analysis of a student's competence but also on their recent behavior related to the constraint.

Our tutorial dialogues not only assist the student to understand the correct problem-solving action for the current context but also help him/her to learn the corresponding domain concept. Thus our dialogues are focused on a student's future performance as well as the current performance. A student's future performance is associated with learning the domain concept associated with the error, reducing the likelihood of repeating the same error. On the other hand current performance focuses on the correct problem-solving action for the current error. We achieve both these goals (problem-solving goals and pedagogical goals) by focusing only on the current error. Our approach is to focus on multiple aspects (reflecting on the current error, discussing the correct problem-solving action, discussing corresponding domain concept and reviewing the domain concept) of the current error. This approach makes it possible for our dialogues to facilitate knowledge construction as well as knowledge remediation.

Developing tutorial dialogues for various error types rather than for each problem makes it possible to add additional problems to the system without requiring any extensions or modifications to the dialogues.

Our adaptive dialogues are designed to maximize learning efficiency by expecting students to go through just one prompt for the first occurrence of an error. The number of prompts a student receives for a certain type of error depends on the number of occurrences for that error within a single session as well as the accuracy of the student's explanations. Learning efficiency is further improved by focusing on the error that a student is most likely to make in subsequent attempts. These attempts to improve learning efficiency are supported by the experimental group learning a significantly higher number of domain concepts (i.e. constraints) while engaging in adaptive dialogues than the control group with non-adaptive dialogues. Both groups spent approximately the same time interacting with dialogue-based EER-Tutor.

We extended constraint-based methodology to provide adaptive tutorial dialogue support in both ill-and well-defined tasks.

5.2 Limitations

There are several limitations of this research which will now be discussed. One of the goals of this research is to propose a model that can provide adaptive dialogue support in multiple domains. As this research was carried out within the context of constraint-based tutoring systems, this model can only be applied to a group of domains where problems allow constraint-based diagnosis. As the error hierarchy is based on the constraint base for the specific domain, the model cannot be developed for that domain until the constraint base is developed. Another limitation was that the model can provide dialogue support only for tutoring systems that provide problem-solving environments. As a consequence, the dialogue support is also limited to a specific kind of dialogue that focuses on discussing errors and the corresponding domain concepts.

The various types of studies that were carried out have their own limitations. The applicability of the model was explored only in four domains (i.e. conceptual database design, logical database design, data normalization and fraction addition). In addition, explorations of two of these domains (i.e. logical database design and fraction addition) were carried out only conceptually and three of them (i.e. conceptual database design, logical database design and data normalization) were closely related. Thus further investigations need to be carried out to evaluate the generality of the model.

The error hierarchies were developed by one researcher and structure of the hierarchy can be influenced by the individual's subjective opinion. Each error hierarchy could have been validated by comparing the exiting one with another that was independently developed. In addition, educational data mining techniques could have been used to validate each error hierarchy. The error hierarchies for three of the domains (i.e. conceptual database design, logical database design, and data normalization) were based on the constraint bases of the existing tutoring systems (i.e. EER-Tutor, ERM-Tutor and NORMIT). So the educational data mining techniques could be used to analyse the student interactions with each of these systems to validate the corresponding hierarchy.

We presented one algorithm to select the pedagogically most suitable error when a student solution contains multiple errors. Our algorithm was based on frequencies, but other algorithms are possible. One possible approach is to use probabilities instead of frequencies. This approach may lead to better selection mechanism.

5.3 Future Directions

This research will pave the way for a number of future explorations. A larger study with dialogue-based NORMIT could be conducted to evaluate the effectiveness of our model in providing dialogue support to learn a well-defined task. We believe one of the major reasons for the non-significant results in the NORMIT study between the pre-test and post-test performances of the two groups, and between the gains is due to the small number of participants (18 vs 65 in EER-Tutor study). However, we observed similar trends in learning in both studies: significantly higher number of constraints learnt in EER-Tutor study, and a higher learning rate in NORMIT study by the respective experimental groups compared to their peers. Thus we believe it will be beneficial to explore the effectiveness of our model providing dialogue support to learn a well-defined task in an authentic classroom environment.

The applicability of our model in an ill-defined task associated with an ill-defined domain theory could be explored. In the first study involving dialogue-based EER-Tutor, the model was used to support dialogues in conceptual database design, an ill-defined task with a well-defined domain theory. In the second study with dialogue-based NORMIT, dialogues facilitated learning data normalization, a well-defined task with a well-defined domain theory. Therefore, our model has shown evidence of enhancing learning for a task in the WDIT quadrant (well-defined domain, ill-defined task) and WDWT quadrant (well-defined domain, well-defined task). Now the next step would be to use the model to support learning a task such as essay writing, an ill-defined task with an ill-defined domain theory. Similar to the directions mentioned above, we can also investigate the applicability of our model in a well-defined task with an ill-defined domain theory such as psychological diagnosis.

The dialogues could be enhanced with natural language capabilities so that students have the opportunity to construct their own explanations. Constructing explanations in one's own words have shown to be more effective than selecting the correct explanation from a list of possible explanations (Aleven & Koedinger, 2002).

Currently, all the dialogue turns are initiated by the system. The model could be extended to provide the opportunity for students to ask questions as they feel necessary. This will provide more opportunities for students to construct their own knowledge.

Our current approach is to increase the number of prompts a learner receives with each repeated mistake. However, it might be beneficial to use a different approach when a student repeatedly makes the same mistake. For instance, if a certain error is repeatedly identified as the one student has most difficulty with, it would be beneficial to engage in a dialogue that discusses general concepts related to the error. For instance, if a student repeatedly makes mistakes about calculating least common denominator (LCD) when adding two fractions, it might be better to focus more on finding the LCD than the problem he/she is attempting to solve (i.e. adding two fractions). It also possible to explore the effectiveness of other pedagogical strategies such as worked-examples for repeated mistakes.

In order to initiate a dialogue, our model selects the error that a student is most likely to make in the subsequent attempts. This has the potential to maximize the learning efficiency. This is supported by the findings of the study involving dialogue-based EER-Tutor. The experimental group learnt a significantly higher number of domain concepts (i.e. constraints) while engaging in adaptive dialogues than the control group with non-adaptive dialogues. Both groups spent approximately the same amount of time interacting with dialogue-based EER-Tutor. However it is interesting to explore whether another approach may be more effective: selecting the error that a student is least likely to make in subsequent attempts. It may assist them to focus on easier errors than harder ones.

Another interesting research question is to explore how the fine-grain analysis of the student's knowledge level provided by the error hierarchy could be used to develop an adaptive problem selection strategy.

A further direction for research is to explore how beneficial would it be to use the detailed view of the student knowledge (provided by the hierarchy) as an open student model. This will provide an opportunity for students to reflect on how their knowledge evolves while learning with the ITS. Accuracy of the explanations provided by a student in response to the dialogue prompts could be used to further support the details of the student model.

Finally, the use of tutorial dialogues in multiple domains could be further facilitated by exploring ways to automate the dialogue authoring process. One possible approach is to investigate how the error hierarchy itself could be used to automate the authoring process.

5.4 Concluding Remarks

As today's educational settings are increasingly driven by technology, the role of intelligent tutoring systems in classrooms have gained prominence. Tutorial dialogues are an effective pedagogical approach used in intelligent tutoring systems to engage the student in a discussion about their problem-solving process. The dialogues also provide opportunities to reflect on the existing knowledge as well as to integrate new knowledge. Dialogues developed in this research focus on knowledge construction as well knowledge remediation i.e. students are asked to engage in dialogues only when their solution is erroneous. While the dialogues assist students to reflect on their mistakes, they are also given an opportunity to learn the corresponding domain concepts they have difficulty with. Even though the dialogues are rich in providing opportunities to learn both procedural and conceptual knowledge, students are not expected to go through the entire dialogue. Instead the dialogues are customised based on each student's knowledge level and explanation skills. As the result, students are encouraged to focus on problem solving while taking advantage of the learning opportunities provided by the adaptive dialogues. In summary, our dialogues are designed to be effective by facilitating both knowledge construction and knowledge remediation within the context of the current problem. On the other hand, they attempt to maximize the learning efficiency by customizing the content and the timing for each individual student. The results provided evidence that our model could be used in multiple domains to teach different types of instructional tasks. Our research has the capability to increase the use of adaptive tutorial dialogues to provide rich learning opportunities in real classroom settings.

References

Aleven, V., & Koedinger, K. (2000). Limitations of Student Control: Do Students Know When They Need Help? In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Proceedings of the Fifth International Conference* (pp. 292-303). Montreal, Canada: Springer-Verlag.

Aleven, V., & Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explain with a computer-based Cognitive Tutor. *Cognitive Science*, *26*(147-179).

Aleven, V., Ashley, K., Lynch, C., & Pinkwart, N. (2006). Proceedings of the workshop on Intelligent Tutoring Systems for Ill-Defined Domains. *8th International Conference on Intelligent Tutoring Systems (ITS 2006)*.

Aleven, V., Ashley, K., Lynch, C., & Pinkwart, N. (2007). Proceedings of the workshop on AIED applications in ill-defined domains. *13th International Conference on Artificial Intelligence in Education (AIED 2007)*.

Aleven, V., Ashley, K., Lynch, C., & Pinkwart, N. (2008). Proceedings of the workshop on Intelligent Tutoring Systems for Ill-Structured Domains Focusing on Assessment and Feedback. *9th International Conference on Intelligent Tutoring Systems (ITS 2008)*.

Aleven, V., Koedinger, K. R., & Popescu, O. (2003). A Tutorial Dialog System to Support Self-Explanation: Evaluation and Open Questions. In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Proceedings of the 11th International onference on Artificial Intelligence in Education, AI-ED 2003* (pp. 39-46). Amsterdam: IOS Press.

Aleven, V., Koedinger, K., & Cross, K. (1999). Tutoring answer explanation fosters learning with understanding. In S. Lajoie & M. Vivet (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 199-206).

Aleven, V., Popescu, O., & Koedinger, K. R. (2002). Pilot-Testing a Tutorial Dialogue System that Supports Self-Explanation. *Proc.eedings of the International Conference on Intelligent Tutoring Systems* (pp. 344-354). Biarritz, France.

Aleven, V., Popescu, O., Ogan, A., & Koedinger, K. R. (2003). A Formative Classroom Evaluation of a Tutorial Dialog System that Supports Self-Explanation. In V. Aleven, U. Hoppe, J. Kay, R. Mizoguchi, H. Pain, F. Verdejo, & K. Yacef (Eds.), *Supplemental Proceedings of 11th International Conf erence on Artificial Intelligence in Education (AIED 2003)*. Sydney, Australia.

Alexe, C., & Gescei, J. (1996). A learning environment for the surgical intensive care unit. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *Third International Conference on Intelligent Tutoring Systems* (pp. 439–447). Montreal, Canada.

Anderson, J. R. (1993). *Rules of the Mind*. (N. L. E. Associates, Ed.).

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. MahWah, NJ, Lawrence Erlbaum Associates.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1996). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, *4*(2), 167–207.

Ayscough, P. B. (1977). "CALCHEMistry." *British Journal of Education Technology*, *8*(3), 201–213.

Azevedo, R., Johnson, A., Chauncey, A., & Burkett, C. (2010). Self-regulated learning with MetaTutor: Advancing the science of learning with MetaCognitive tools. In M. Khine & I. Saleh (Eds.), *New science of learning: Computers, cognition, and collaboration in education* (pp. 225-247). Amsterdam: Springer.

Baghaei, N., Mitrovic, A., & Irwin, W. (2007). Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. *International Journal of Computer Supported Collaborative Work*, *2*(2-3), 159-190.

Bartini, C., Lenzerini, M., & Navathe, S. B. (1986). A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, *18*(2), 323-364.

Bloom, B. S. (1984). The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*, 4-16.

Brusilovsky, P. L. (2000). Adaptive Hypermedia: From Intelligent Tutoring Systems to Web-Based Education. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Proceedings of the Fifth International Conference on Intelligent Tutoring Systems* (pp. 1-7). Montreal, Canada: Springer-Verlag.

Brusilovsky, P., & Weber, G. (2001). ELM-ART: an adaptive versatile system for Web-based instruction. *International Journal of Artificial Intelligence in Education*, *12*, 351-384.

Brusilovsky, P., Karagiannidis, C., & Sampson, D. (2001). The benefits of layered evaluation of adaptive applications and services. *Proceedings of the 1st Workshop on Empirical Evaluation of Adaptive Systems at User Modelling Conference (UM2001)* (pp. 1-8). Sonthofen, Germany.

Bunt, A., Conati, C., & Muldner, K. (2004). Scaffolding self-explanation to improve learning in exploratory learning environments. *7th International Conference on Intelligent Tutoring Systems* (pp. 656-667). Lecture Notes in Computer Science, Volume 3220/2004, Springer Berlin, Heidelberg,.

Chi, M. T. H. (2000). Self-explaining expository texts: The dual processes of generating inferences and repairing mental models. *Advances in Instructional Psychology*, 161-238.

Chi, M. T. H., Siler, S. A., Jeong, H., Yamauchi, T., & Hausmann, R. . (2001). Learning from human tutoring. *Cognitive Science, 25*(471-533).

Chi, M., VanLehn, K., Litman, D., & Jordan, P. (2011a). Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction. Special Issue on Data Mining for Personalized Educational Systems, 21*(1-2), 137-180.

Chi, M., VanLehn, K., Litman, D., & Jordan, P. (2011b). An Evaluation of Pedagogical Tutorial Tactics for a Natural Language Tutoring System: A Reinforcement Learning Approach. *Special Issue on Best of ITS 2010, International Journal of Artificial Intelligence in Education, 21*(1-2), 83-113.

Corbett, A. T., Trask, H. J., Scarpinatto, K. C., & Hadley, W. S. (1998). A formative evaluation of the PACT Algebra II tutor: Support for simple hierarchical reasoning. In B. P. Goettl, H. M. Halff, C. L. Redfield, & V. J. Shute (Eds.), *4th International Conference on Intelligent Tutoring Systems* (pp. 374–383). San Antonio, Texas.

Dillenbourg, P., & Self, J. A. (1992). People power: a human-computer collaborative learning system. In C. Frasson, G. Gauthier, & G. McCalla (Eds.), *Second International Conference on Intelligent Tutoring Systems* (pp. 651-660). Montreal, Canada: Springer-Verlag.

Elmasri, R., & Navathe, S. B. (2010). *Fundamentals of Database Systems* (6th ed.). Addison-Wesley.

Evans, M., & Michael, J. (2006). *One-on-one Tutoring by Humans and Computers*. Lawrence Erlbaum Associates Inc.

Forbes-Riley, K., & Litman, D. (2011). When Does Disengagement Correlate with Learning in Spoken Dialog Computer Tutoring? *15th Interbnational Conference on Artificial Intelligence in Education* (pp. 81-89). Auckland, NZ: Springer-Verlag Berlin Heidelberg. doi:Lecture Notes in Computer Science Volume 6738

Freedman, R., Rose, C. P., Ringenberg, M. A., & VanLehn, K. (2000). ITS tools for natural language dialogue: A domain-independent parser and planner. *Proceedings of the*

International Conference on Intelligent Tutoring Systems (pp. 433-442). Montreal:Canada: Springer-Verlag Berlin & Heidelberg.

Gertner, A., & VanLehn, K. (2000). ANDES: A Coached Problem-Solving Environment for Physics. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 133-142). New York : Springer.

Goel, V., & Pirolli, P. (1993). The structure of design problem spaces. *Cognitive Science, 16*, 395-429.

Graesser, A. C. (2011). Learning, thinking, and emoting with discourse technologies. *American Psychologist, 66*, 743-757.

Graesser, A. C., D'Mello, S. K., & Person, N. K. (2009). Meta-knowledge in tutoring. In D. J. Hacker, J. Dunlosky, & A. C. Graesser (Eds.), *Handbook of metacognition in education*. Mahwah, NJ: Erlbaum.

Graesser, A. C., Franceschetti, D. R., Gholson, B., & Craig, S. D. (2011). Learning Newtonian physics with conversational agents and interactive simulation. In N. L. Stein & S. Raudenbush (Eds.), *Developmental cognitive science goes to school* (pp. 157-172). New York: Routledge.

Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., & Olney, A. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavioral Reseach Methods, Instruments and Computers, 36*, 180–193.

Graesser, A. C., Moreno, K., Marineau, J., Adcock, A., Olney, A., & Person, N. (2003). AutoTutor improves deep learning of computer literacy: Is it the dialog or the talking head? In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 47-54). Amsterdam: IOS Press.

Graesser, A. C., Rus, V., D'Mello, S. K., & Jackson, G. T. (2008). AutoTutor: Learning through natural language dialogue that adapts to the cognitive and affective states of the learner. In D. H. Robinson & G. Schraw (Eds.), *Recent innovations in educational technology that facilitate student learning* (pp. 95-125). Information Age Publishing.

Graesser, A. C., VanLehn, K., Rosé, C., Jordan, P., & Harter, D. (2001). Intelligent Tutoring Systems with Conversational Dialogue. *AI Magazine, 22*(4), 39-51.

Heffernan, N. T., & Croteau, E. (2004). Web-Based Evaluations Showing Differential Learning for Tutorial Strategies Employed by the Ms. Lindquist Tutor. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 491-500). Maceio, Brazil: Springer-Verlag.

Heffernan, N. T., Koedinger, K. R., & Razzaq, L. (2008). Expanding the Model-Tracing Architecture: A 3rd Generation Intelligent Tutor for Algebra Symbolization. *International Journal of Artificial Intelligence in Education, 18*(2), 153-178.

Holt, P., Dubs, S., Jones, M., & Greer, J. (1994). The State of Student Modelling. In J. Greer & G. McCalla (Eds.), *Student Modelling: Key to Individualized Instruction* (pp. 1-35). New York: Springer Verlag.

Hu, X., & Graesser, A. C. (2004). Human use regulatory affairs advisor (HURAA): Learning about research ethics with intelligent learning modules. *Behavior Research Methods, Instruments & Computers*, *36*(2), 241-249.

Hussain, M. S., AlZoubi, O., Calvo, R. A., & D'Mello, S. (2011). Affect Detection from Multichannel Physiology during Learning Sessions with AutoTutor. *15th Interbnational Conference on Artificial Intelligence in Education* (pp. 131-138). Auckland, NZ: Springer-Verlag Berlin Heidelberg.

Jordan, P., Makatchev, M., Pappuswamy, U., VanLehn, K., & Albacete, P. (2006). A natural language tutorial dialogue system for physics. In G. Sutcliffe & R. Goebel (Eds.), *Proceedings of the 19th International FLAIRS Conference*. Menlo Park, CA: AAAI Press.

Jordan, P., Ringenberg, M., & Hall, B. (2006). Rapidly Developing Dialogue Systems that Support Learning Studies. *Proceedings of ITS06 Workshop on Teaching with Robots, Agents, and NLP* (pp. 29-36).

Koedinger, K., & Aleven, V. (2007). Exploring the assistance dilemma in experiments with Cognitive Tutors. *Educational Psychology Review*, *19*(3), 239-264.

Koedinger, K., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, *8*, 30-43.

Kulik, J. A., Kulik, C.-L. C., & Cohen, P. A. (1980). Effectiveness of computer-based college teaching: a meta-analysis of findings. *Review of Educational Research, 50*(4), 524–544.

Kung, H.-J., & Tung, H.-L. (2006). An alternative approach to teaching database normalization: A simple algorithm and an interactive e-Learning tool. *Journal of Information Systems Education, 17*(30), 315-325.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 211--240.

Last, R. W. (1979). The role of computer-assisted learning in modern language teaching. *Association for Literary and Linguistic Computing, 7*, 165 - 171.

Lehman, B. A., D'Mello, S. K., Strain, A., Gross, M., Dobbins, A., Wallace, P., Millis, K., et al. (2011). Inducing and Tracking Confusion with Contradictions during Critical Thinking and Scientific Reasoning. *15th International Conference on Artificial Intelligence in Education* (pp. 171-178). Auckland, NZ: Springer-Verlag Berlin Heidelberg.

Lepper, M. R., Woolverton, M., Mumme, D. L., & Gurtner, J. L. (1993). No Title. In S. P. Lajoie & S. J. Derry (Eds.), *Computer as Cognitive Tools* (pp. 75-105). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Lesgold, A. (1987). Toward a theory of curriculum for use in designing intelligent instructional systems. In H. Mandl & A. M. Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems* (pp. 114–137). Springer-Verlag New York.

MacGregor, R. (1991). The Evolving Technology of Classification-Based Knowledge Representation Systems. In J. Sowa (Ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. San Mateo, CA: Morgan Kaufmann.

Makatchev, M., Hall, B., Jordan, P., Pappuswamy, U., & VanLehn, K. (2005). Mixed language processing in the Why2-Atlas tutoring system. *Proceedings of the Workshop on Mixed Language Explanations in Learning Environments, held in conjunction with 12th International Conference on Artifical Intelligence in Education*. Amsterdam, Netherlands.

Martin, B., & Mitrovic, A. (2002). Authoring web-based tutoring systems with WETAS. In Kinshuk, R. Lewis, K. Akahori, R. Kemp, T. Okamoto, L. Henderson, & C.-H. Lee (Eds.), *International Conference on Computers in Education* (pp. 183–187). Auckland, NZ.

Mayo, M., Mitrovic, A., & McKenzie, J. (2002). CAPIT: An intelligent tutoring system for capitalisation and punctuation. In Kinshuk, C. Jesshope, & T. Okamoto (Eds.), *Advanced Learning Technology: Design and Development Issues, IEEE Computer Society* (pp. 151–154). Los Alamitos, CA.

Michael, J., Rovick, A., Glass, M., Yujian, Z., & Evens, M. (2003). Learning from a Computer Tutor with Natural Language Capabilities. *Interactive Learning Environments*, *11*(3), 233-262.

Milik, M., Marshall, M., & Mitrovic, A. (2006). Teaching logical database design in ERM-Tutor. In M. Ikeda & K. Ashley (Eds.), *International Conference on Intelligent Tutoring Systems* (pp. 707-709). Berlin Springer.

Millis, K., Forsyth, C., Butler, H., Wallace, P., Graesser, A. C., & Halpern, D. (2011). Operation ARIES! A serious game for teaching scientific inquiry. In M. Ma, A. Oikonomou, & J.

Lakhmi (Eds.), *Serious Games and Edutainment Applications* (pp. 169-195). Springer-Verlag: London.

Mitrovic, A. (1998). Experiences in Implementing Constraint-Based Modelling in SQL Tutor. In B. P. Goettl, H. M. Halff, C. L. Redfield, & V. J. Shute (Eds.), *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems* (pp. 414-423). San Antonio, Texas: Springer-Verlag.

Mitrovic, A. (2005). The Effect of Explaining on Learning: a Case Study with a Data Normalization Tutor. *12th International Conference on Artificial Intelligence in Education* (pp. 499-506). Amsterdam, The Netherlands.

Mitrovic, A. (2012). Fifteen years of Constraint-Based Tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, *22*((1-2)), 39-72.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a Constraint-Based Tutor for a Database Language. *International Conference on Artificial Intelligence in Education*, *10*((3-4)), 238-256.

Mitrovic, A., & Weerasinghe, A. (2009). Revisiting Ill-Definedness and the Consequences for ITSs. *Proceedings of the 14th International Conference on Artificial Intelligence in Education* (pp. 375-382). Brighton, UK.

Mitrovic, A., Martin, B., & Mayo, M. (2002). Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, *12*((2-3)), 243-279.

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent Tutors for All: The Constraint-Based modeling methodology, systems and authoring. *IEEE Intelligent Systems, special issue on Intelligent Educational Systems*, *22*(4), 38-45. Published by the IEEE Computer Society. doi:10.1109/MIS.2007.74

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., & McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education*, *19*(2), 155-188.

Mitrovic, A., Mathews, M., & Holland, J. (2012). Exploring two strategies for teaching procedures. *11th International Conference on Intelligent Tutoring Systems* (p. Accepted for publication).

Mitrovic, Antonija. (2001). Investigating students' self-assessment skills. *User Modeling 2001* (pp. 247-250). Retrieved from http://www.springerlink.com/index/3pq6d6q8vphg236u.pdf

Munro, A., Johnson, M. C., Pizzini, Q. A., Surmon, D. S., Towne, D. M., & Wogulis, J. L. (1997). Authoring simulation-centred tutors with RIDES. *International Journal of Artificial Intelligence in Education, 8*, 284–316.

Ohlsson, S. (1992). Constraint-based student modelling. *International Journal of Artificial Intelligence in Education, 3*(4), 429-477.

Ohlsson, S. (1994). Constraint-based student modelling. *Student Modelling: the Key to Individualized Knowledge-based Instruction* (pp. 167–189). Springer-Verlag, Berlin.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review, 103*(2), 241–262.

O' Shea, T., & Self, J. A. (1983). *Learning and Teaching with Computers*. Brighton, Harvester Press.

Paramythis, A., Totter, A., & Stephanidis, C. (2001). A modular approach to the evaluation of adaptive user interfaces. *Proceedings of the 1st Workshop on Empirical Evaluation of Adaptive Systems at User Modelling Conference (UM2001)* (pp. 9–24). Sonthofen,Germany.

Paramythis, Alexandros, Weibelzahl, S., & Masthoff, J. (2010). Layered evaluation of interactive adaptive systems: framework and formative methods. *User Modeling and User-Adapted Interaction, 20*(5), 383-453.

Person, N. K., Graesser, A. C., Bautista, L., Mathews, E. C., & The Tutoring Research Group. (2001). Evaluating student learning gains in two versions of AutoTutor. In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 286–293). Amsterdam: IOS Press.

Phillip, G. C. (2007). Teaching database modeling and design: areas of confusion and helpful hints. *Journal of Information Technology Education, 6*, 481-497.

Reitman, W. R. (1964). Heuristic Decision Procedures, Open Contraints , and the Structure of Ill-defined Problems. In M. W. Shelly & G. L. Bryan (Eds.), *Human judgements and optimality* (pp. 282-315).

Rich, E. (1989). Stereotypes and User models. *User Models in Dialog Systems*, 35–51.

Rose, C. P., Bhembe, D., Siler, S., Srivastava, R., & VanLehn, K. (2003). Exploring the Effectiveness of Knowledge Construction Dialogues. *Proceedings of the International Conference on Artificial Intelligence in Education.* Amsterdam: IOS Press.

Rosé, C. P., & Lavie, A. (1999). LCFlex: An Efficient Robust Left-Corner Parser.

Rosé, C P, Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive conceptual tutoring in Atlas-Andes. *Proceedings of AI in Education 2001 Conference* (pp. 151-153). Retrieved from http://www.public.asu.edu/~kvanlehn/Stringent/PDF/01AIED_CR_PJ_MR_SS_KVL_AW.pdf

Rosé, C., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive conceptual tutoring in Atlas-Andes. In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *International Conference on Artificial Intelligence in Education* (pp. 256-266). Amsterdam: IOS Press.

Shute, V. J., Glaser, R., & Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In P. Ackerman, R. Sternberg, & R. Glaser (Eds.), *Learning and individual differences: Advances in theory and research* (pp. 279–326). New York Freeman.

Siler, S., Rosé, C., Frost, T., VanLehn, K., & Koehler, P. (2002). Evaluating knowledge construction dialogs (KCDs) versus minilessons within Andes2 and alone. *ITS2002 Workshop on Empirical Methods for Tutorial Dialogue Systems* (pp. 9-15). San Sebastian, Spain.

Suraweera, Pramuditha. (2001). *An Intelligent Teaching System for Database Modelling*. University of Canterbury.

Suraweera, Pramuditha, & Mitrovic, A. (2004). An Intelligent Tutoring System for Entity Relationship Modelling. *International Journal of Artificial Intelligence in Education*, *14*(3-4), 375-417.

Taylor, R. N. (1974). Nature of problem ill-structuredness: implications for problem formulation and solution. *Decision Sciences*, *5*, 632-643.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems and other tutoring systems. *Educational Psychologist*, *46*(4), 197-221.

VanLehn, K., Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rose, C., et al. (2000). Fading and deepening: The next steps for Andes and other model-tracing tutors. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.), *Proceedings of the 5th International Conference on Intelligent Tutoring Systems (ITS 2000)* (pp. 474-483). Berlin: Springer Verlag.

VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rosé, C. P. (2007). When are tutorial dialogues more effective than reading? *Cognitive Science*, *31*(1), 3-62. Lawrence Erlbaum Associates, Inc. 10 Industrial Avenue Mahwah, NJ 07430-2262 USA. doi:10.1080/03640210709336984

VanLehn, K., Jordan, P. W., Rose, C. P., Bhembe, D., Boettner, M., Gaydos, A., Pappuswamy, M. M. U., et al. (2002). The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing. In S. A. Cerri, G. Gouraderes, & F. Paraguacu (Eds.), *In Proceedings of the International Intelligent Tutoring Systems Conference* (pp. 158-167). Berlin, Germany: Springer.

VanLehn, K., Siler, S., Murray, C., & Baggett, W. (1998). What makes a tutorial event effective? In M. A. Gernsbacher & S. Derry (Eds.), *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society* (pp. 1084-1089). Hillsdale, NJ: Erlbaum.

VanLehn, K., van de Sande, B., Shelby, R., & Gershman, S. (2010). The Andes physics tutoring system: An experiment in freedom. In R. Nkambou & J. Bourdeau (Eds.), *Advances in Intelligent Tutoring Systems* (pp. 421-446). Springer-Verlag, Berlin.

Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. (M. Cole, V. John-Steine, S. Scribner, & E. Souberman, Eds.).

Weerasinghe, A. (2003). *Exploring the effects of self-explanation in the context of a database design tutor*. University of Canterbury.

Weerasinghe, A., & Mitrovic, A. (2006). Facilitating Deep Learning Through Self-Explanation in an Open-ended Domain. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, *10*(1), 3-19.

Weibelzahl, S. (2001). Evaluation of adaptive systems. *Proceedings of the 8th International Conference on User Modelilng* (pp. 292–294). Springer-Verlag, Berlin.

Appendix A

Error Hierarchies for Different Types of Domains

A.1 Conceptual Database Design

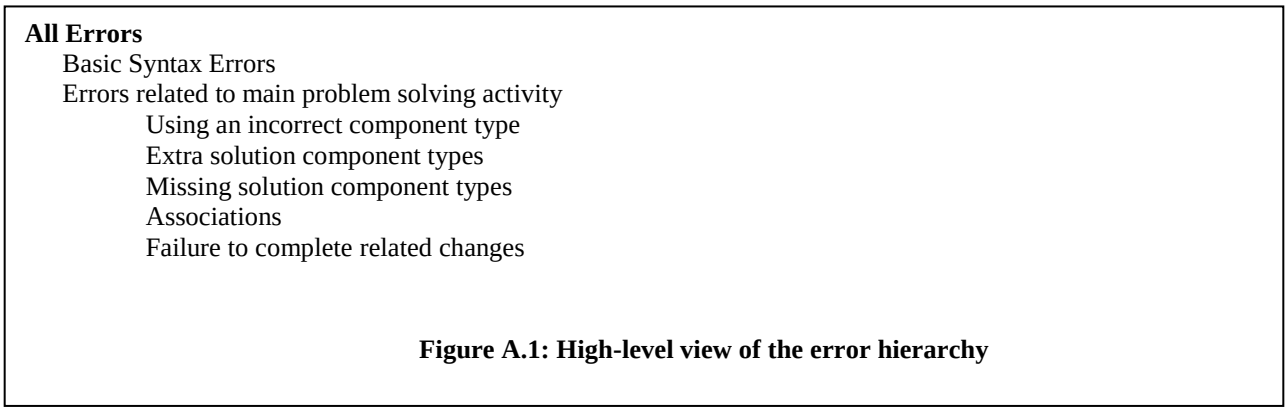


Figure A.1: High-level view of the error hierarchy

The high-level view (Figure A.1) is the same for all the hierarchies presentedhere.

- **Basic Syntax Errors**
  - ER
    - Empty solution
      - 0
    - Connectors
      - Simple connections
        - 3
      - Entity
        - 7
      - Relationships
        - 8
        - 21
      - Attributes
        - 3_11 or 3_12
        - 4
    - Tags
      - Entity
        - 1
        - 2_A
      - Relationships
        - Non-recursive relationship tags
          - 1_A
          - 2_B
        - Recursive relationship tags
          - 21-AR
      - Attributes
        - 1_B
        - 2_C
    - Unique names
      - Entity
        - 10
        - 12
      - Relationship
        - Non-recursive relationship names
        - 11_A
        - Recursive relationship names
        - 80
      - Attributes
        - 10_A
        - 10_B
  - EER
    - Connectors
      - Subset
        - 90
      - Non-subset
        - 87
        - 89
      - Direction
        - 91
      - Junction
        - 88
    - Specialisations
      - Non-single
        - 101
      - Single
        - 100
        - 102
      - Cardinality
        - 103
    - Categories
      - Non-cardinality
        - 120
      - Cardinality
        - 121

Figure A.2: Detailed view of the node *Basic Syntax Errors*

Using an incorrect solution component type

- ER Constructs
  - Using a completely different type of solution component
    - Using an entity to represent another type of solution component
      - Using an entity to represent a relationship
        - 27 or 28
      - Using an entity to represent an attribute
        - 202 or 203 or 204 or 205 or 206 or 202-1or 205-1or 206-1
    - Using a relationship to represent another type of solution component
      - Using a relationship to represent an entity
        - 13_1or 14_1
      - Using a relationship to represent an attribute
        - 207 or 208 or 209 or 210 or 211_A or 207_1 or 210_1 or 211_B
    - Using an attribute to represent another type of solution component
      - Using an attribute to represent an entity
        - 13_2 or 14_2
        - 65-5
        - 65-6
      - Using an attribute to represent a relationship
        - 27_2 or 28-2
  - Using a different variation of the correct solution component
    - Entity
      - Using a regular entity to represent a weak entity
        - 14
        - 67-2
      - Using a weak entity to represent a regular entity
        - 13
    - Relationship
      - Using a regular relationship to represent an identifying relationship
        - 28_1
      - Using an identifying relationship to represent a regular relationship
        - 27-1
    - Attribute
      - Using a different type of complete attribute
        - 54 or 54_1 or 55 or 56 or 57 or 57_1 or 58 or 58_1 or 59 or 59_1
        - 65-2
      - Using a different type of component attribute
        - 41 or 42
- EER Constructs
  - Using a completely different type of solution component
    - Sub class
      - Using a weak entity to represent a sub class
        - 115
    - Category
      - Using a weak entity to represent a category
        - 130
  - Using a different variation of the correct solution component
    - Sub class
      - Sub class doubling as a category
        - 113
    - Category
      - Category doubling as a sub class
        - 128

Figure A.3: Detailed view of the node *Using an incorrect solution component type*

Extra solution component types

- Entity
  - Regular
    - 199
  - Weak
    - 201 OR 201_1
- Relationships
  - Regular
    - 205_A
  - Identifying
    - 206_A or 206_A1

Figure A.4: Detailed view of the node *Extra solution component types*

Missing solution component types

- Entity
  - Regular
    - 15
  - Weak
    - 16
    - 65_1
  - Relationship
    - 37
  - Attributes
    - Key attributes
      - 61
      - 62
    - Non-key attributes
      - 60 OR 60_1
      - 63 OR 63_1
      - 67_1

Figure A.5: Detailed view of the *node Missing* solution component types

Associations

- ER Constructs
  - Associations between different types of solution components
    - Entities and relationships
    - Entities and attributes
    - Relationships and attributes
  - Associations between same types of solution components
    - Attributes and its components
- EER Constructs
  - Associations between different types of solution components
    - Classes and attributes
    - Categories and attributes
  - Associations between same types of solution components
    - Other connections
    - Constraints for categories and specialisation

Figure A.6: Detailed view of the node *Associations*

Entities and Relationships

- Non-recursive
  - Non-structural constraints
    - Extra participating component types
      - Entity
        - Regular
          - 29
        - Weak
          - 31
        - Owner
          - 30 or 67_3
    - Missing participating component types
      - Entity
        - Regular
          - 26
        - Weak
          - 22
        - Owner
          - 26_1
          - 23
      - Relationship
          - 25
  - Structural constraints
  - Cardinality
      - Regular relationship
        - Incorrect value
          - 92 or 94
        - No-value
          - 92_1 or 94_1
      - Identifying relationship
        - Weak entity
          - 92_W or 94_W
        - Owner entity
          - 92_O or 33
    - Participation
      - Regular relationship
        - 96 or 98
      - Identifying relationship
        - Weak entity
          - 34
        - Owner entity
          - 96_O or 98_O
- Recursive
  - Non-structural
  - Extra participating component types
      - 21_B or 78
    - Missing participating component types
      - 21_A
  - Structural
  - Cardinality
      - 92_R or 94_R
    - Participation
      - 96_R or 98_R
    - Role names
      - 81 or 82 or 83

Figure A.7: Detailed view of the node *Entities and Relationships*

Entities and attributes

Extra participating attributes

49 or 50 or 51 or 52 or 53 or 60_A or 67_4

76

46_1

47

48

Missing participating attributes

60_B

64 or 64_1 or 67_5

44

45

46

Figure A.8: Detailed view of the node *Entities and attributes*

Relationships and attributes

Extra participating attributes

49_1 or 52_1 or 53_1 or 60_1A

77

72 or 72_a

73 or 73_a

74

75

Missing participating attributes

64_A or 64_1A

Figure A.9: Detailed view of the node *Relationships and attributes*

Attributes and its components

Extra participating component types

All components

69 or 69_1 or 69_2

39 or 39_1

Some components

70 or 65_3

5

Missing participating component types

All components

64_2 or 64_2A

68 or 68_1 or 68_2

Some components

79 or 71 or 65_4

40 or 40_A

Figure A.10: Detailed view of the node*Attributes and its components*

Classes and attributes

Extra participating attributes

116

Missing participating attributes

117

Figure A.11: Detailed view of the node *Classes and attributes*

Categories and attributes

Extra participating attributes

131

Missing participating attributes

132

Figure A.12: Detailed view of the node *Categories and attributes*

Categories and specialisations

Disjointness /Overlapping

164

Participation

165

Completeness

168

Figure A.13: Detailed view of the node *Categories and specialisations*

Other Associations

- Between super classes and subclasses
  - Extra participating solution components
    - Superclass
      - 162
      - 111
    - Subclass
      - 163
      - 166
  - Missing participating solution components
    - Superclass
      - 160
      - 110
    - Subclass
      - 161
      - 112
- Between categories and super classes
  - Extra participating solution components
    - Category superclass
      - 171
    - Category
      - 172
      - 126
  - Missing participating solution components
    - Category superclass
      - 169
      - 125
      - 133
    - Category
      - 170
      - 127

Figure A.14: Detailed view of the node *Other Associations*

Failure to complete related changes

- Entity
  - Change regular entity to weak entity
    - Changes in the relationships
      - 300
    - Changes in the attributes
      - 301
    - Changes in participation
      - 302
    - Changes in cardinality
      - 303
  - Change weak entity to regular entity
    - Changes in relationships
      - 304
    - Changes in the attributes
      - 305
- Relationship
  - Change regular relationship to an identifying relationship
    - 306
  - Change identifying relationship to regular relationship
    - 307
- Attributes
  - Change key attribute to partial key attribute
    - 308
  - Change partial key attribute to key attribute
    - 309

Figure A.15: Detailed view of the node *Failure to complete related changes*

A.2 Data Normalisation

Basic Syntax Errors
- Attributes being null
- Validity of attributes
- Table names
- Not specifying normal forms
- Specifying higher normal forms before lower normal forms
- Duplicate components

Figure A.16: Detailed view of the node *Basic Syntax Errors*

Attributes being null
- Closure
  - 1
  - 2
- Candidate keys
  - 8
- Prime attributes
  - 13
- Simplify functional dependencies
  - 25
- Partial functional dependencies
  - 98
- Functional dependencies that violate 3NF
  - 34
- Functional dependencies that violate BCNF
  - 38
- Reduce left hand side
  - 55
- Minimal cover
  - 60
- Decomposition
  - 73

Figure A.17: Detailed view of the node *Attributes being null*

Validity of attributes

Closure

3

4

Candidate keys

9

Prime attributes

14

Simplify functional dependencies

41

Partial functional dependencies

42

Functional dependencies that violate 3NF

46

Functional dependencies that violate BCNF

47

Reduce left hand side

56

Minimal cover

61

Decomposition

74

Figure A.18: Detailed view of the node *Validity of attributes*

Table names

No table name

75

Non-unique

76

Figure A.19: Detailed view of the node *Table names*

Not specifying normal forms

1NF

20

2NF

23

3NF

31

BCNF

36

Figure A.20: Detailed view of the node *Not specifying normal forms*

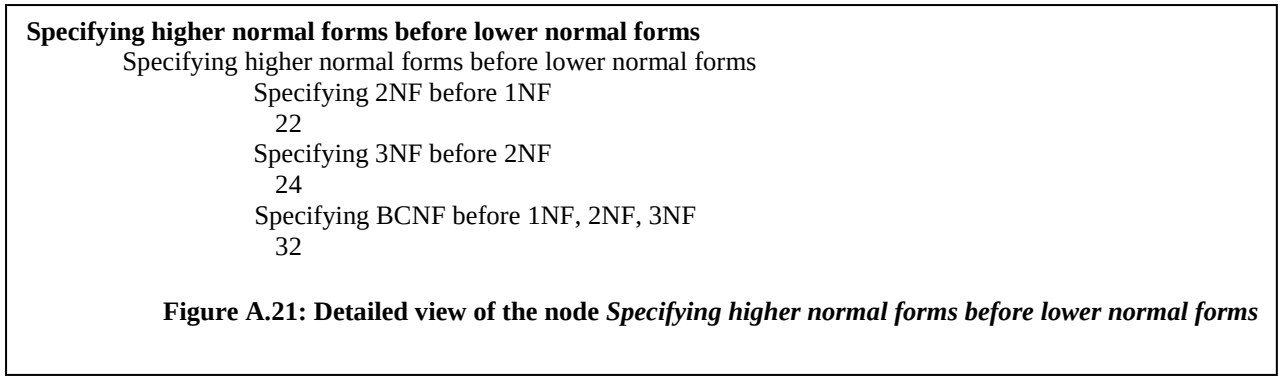


Figure A.21: Detailed view of the node *Specifying higher normal forms before lower normal forms*

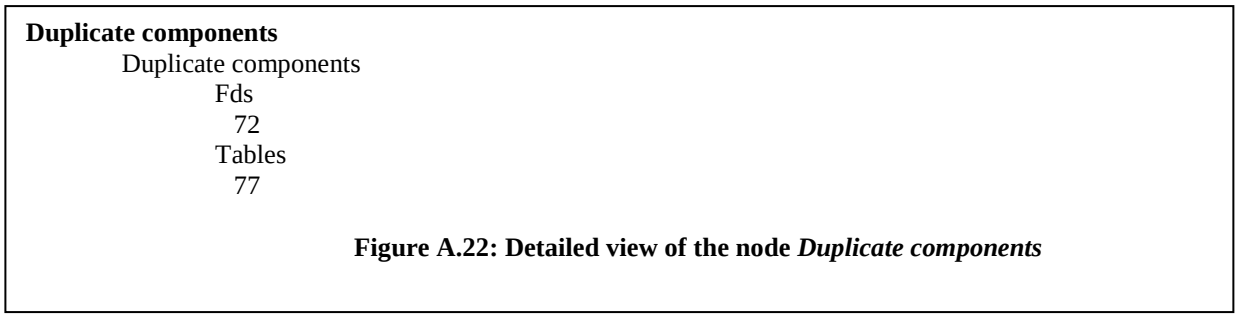


Figure A.22: Detailed view of the node *Duplicate components*

Using an incorrect solution component type

- Candidate keys
  - Specifying a non-super key as a candidate key
    - 10
  - Specifying a non-minimal key as a candidate key
    - 11
- Prime attributes
  - Specifying a non-prime attribute as a prime attribute
    - 15
- Functional dependencies
  - Identifying simple-functional dependencies (fds) as non-simple fds
    - 58
- Normal forms
  - Specifying non-violating fds as violating normal forms
    - 2NF
      - 29
    - 3NF
      - 35
    - BCNF
      - 39
  - Specifying a table not to be in a normal form when it is
    - 1NF
      - 21
    - 2NF
      - 26
      - 27
      - 30
  - Specifying a table to be in a normal form when it is not
    - 2NF
      - 40
    - 3NF
      - 33
    - BCNF
      - 37

Figure A.23: Detailed view of the node *Using an incorrect solution component type*

Extra solution components

- Attributes
  - 82
- Functional dependencies
  - Extra trivial functional dependencies
    - Simplifying functional dependencies
      - 52
      - 53
    - Reduce LHS
      - 65
      - 66
    - Minimal cover
      - 67
      - 68
    - Minimal cover
      - 62
      - 69
    - Redundant functional dependencies
      - 70
      - 71

Figure A.24: Detailed view of the node *Extra solution components*

Missing solution components

- Candidate keys
  - 12
- Prime attributes
  - 16
- Functional dependencies
  - Simplifying functional dependencies
    - 44
    - 45
  - Functional dependencies that violate normal forms
    - 2NF
      - 43
    - 3NF
      - 49
    - BCNF
      - 51
  - Reduce LHS of functional dependencies
    - 57
- Table
  - Missing table for LHS of a functional dependencies
    - 78
  - Missing components of table
    - Missing attributes for RHS of a functional dependency
      - 79
    - Missing key
      - 80

Figure A.25: Detailed view of the node *Missing solution components*

Associations

Closure

Extra participating attributes

Attributes included in closure but not dependent on attributes -set

6

Missing participating attributes

Missing attributes based on reflexive rule

5

Missing right-hand side attributes when left-hand side attributes are in closure

7

Using the decomposition rule incorrectly

Including incorrect attributes in modified functional dependencies

54

Including more than one attribute in RHS

Simplifying functional dependencies

18

Reduce LHS

63

Minimal cover

64

Reducing LHS of functional dependencies incorrectly

59

81

Figure A.26: Detailed view of the node *Associations*

A.3 Logical Database Design

Basic Syntax Errors

- Table name null
  - 101,202, 306, 406,506,705
- Identifying relationship null
  - 205
- Table name not null
  - 201
- Relationships not null
  - 1:1
    - 3011
  - 1:N
    - 4011
  - M:N
    - 5011
  - Multi-valued attributes
    - 6011
  - n-ary relationship
    - 7011
- Relationship null
  - 1:1
    - 301
  - 1:N
    - 401
  - M:N
    - 501
  - Multi-valued attribute
    - 601
  - n-ary relationship
    - 701
- Table empty
  - 104,208
- Invalid attribute name
  - No key
    - 111,214
  - No foreign key
    - 215, 310,410,604
- Foreign key not null
  - 115

Figure A.27: Detailed view of the node *Basic Syntax Errors*

Missing solution component type

- Entity
  - Regular
    - 116
  - Weak
    - 225
- Relationship
  - 1:1 relationship
    - 318
  - 1:N relationship
    - 418
  - M:N relationship
    - 515
  - N-ary relationship
    - 714
- Attributes
  - Keys
    - 220,607,608
    - 609
  - Foreign keys
    - 223, 311,411,507, 508, 706,707
  - Partial keys
    - 224
  - Multi-valued attribute
    - 612
  - Other
    - 108,211
    - 110,213
    - 316,416
    - 513
    - 610

Figure A.28: Detailed view of the node *Missing solution component type*

Using an incorrect solution component type

Using a weak entity instead of a regular entity
103

Using a regular entity instead of a weak entity
204

Using a regular relationship instead of an identifying relationship
206

Using an identifying relationship instead of a regular relationship
303,405,503,704, 303, 405,503, 704

Using a binary relationship instead of a non-binary relationship
304,403,504

Using a non 1:1 relationship instead of a 1:1 relationship
305

Using a non 1:N relationship instead of 1:N relationship
404

Using a 1:N relationship instead of a M:N relationship
505

Using a non n-ary relationship instead of a n-ary relationship
703

Using a composite attribute instead of its components
106, 210

Using a non-candidate key as a candidate key
114

Using a non-primary key as a foreign key
217,314,315, 414,415,511,512,606,710,711

Using a multi-valued attribute instead of a non multi-valued attribute
603

Figure A.29: Detailed view of the node *Using an incorrect solution component type*

Extra solution component type

- Extra table mapped
  - 102,203,307,407
- Invalid relationship name
  - 1: 1
    - 302
  - 1: N
    - 402
  - M: N
    - 502
  - Multi-valued attribute
    - 602
  - n-ary relationship
    - 702
- Attributes
  - 105,209
  - 109, 212
  - 116,225
  - 317,417
  - 514
  - 611
- Keys
  - Primary keys
    - 112, 218,219,221,222, 308,409
  - Foreign keys
    - 218, 312,412,712,713

Figure A.30: Detailed view of the node *Extra solution component type*

Associations

- Extra participating attributes
  - Simple attributes
    - 107,110
  - Key attributes
    - 113, 216
  - Foreign key attributes
    - 313,413,509,510,605,708,709
- Extra participating relationship
  - 207
- Extra participating entity
  - 408

Figure A.31: Detailed view of the node *Associations*

A.4 Fraction Addition

Basic Syntax Errors
- Null value
- Value less than zero
- Value not being an integer

Figure A.32: Detailed view of the node *Basic Syntax Errors*

Null value
- LCD
  - 0_gsy
- Improper Fraction
  - Fraction 1
    - Numerator
      - 3_gsy
    - Denominator
      - 5_gsy
  - Fraction 2
    - Numerator
      - 8_gsy
    - Denominator
      - 10_gsy
  - Sum
    - Numerator
      - 13_gsy
    - Denominator
      - 15_gsy

Figure A.33: Detailed view of the node *Null value*

Value less than zero
- LCD
  - 1_gsy
- Improper Fraction
  - Fraction 1
    - 6_gsy
  - Fraction 2
    - 11_gsy
- Sum
  - 16_gsy
- Reduced Sum
  - 19_gsy

Figure A.34: Detailed view of the node *Value less than zero*

Value not being an integer

- LCD
  - 2_gsy
- Improper Fraction
  - Fraction 1
    - Numerator
      - 4_gsy
    - Denominator
      - 7_gsy
  - Fraction 2
    - Numerator
      - 9_gsy
    - Denominator
      - 12_gsy
- Sum
  - Numerator
    - 14_gsy
  - Denominator
    - 17_gsy
- Reduced Sum
  - Numerator
    - 18_gsy
  - Denominator
    - 20_gsy

Figure A.35: Detailed view of the node *Value not being an integer*

Using an incorrect solution component type

- Using an incorrect least common denominator (LCD)
  - Higher multiple of the correct LCD
    - 22-gse
  - Incorrect LCD
    - 21-gse
    - 0-gse
- Using an incorrect denominator
  - Fraction1
    - 25-gse
  - Fraction 2
    - 26-gse
  - Sum
    - 27-gse
- Using an incorrect whole number
  - 31-gse

Figure A.36: Detailed view of the node *Using an incorrect solution component type*

Associations

- Fratction1
  - Numerator
    - 23-gse
- Fratction2
  - Numerator
    - 24-gse
- Sum
  - Numerator
    - 28-gse
- Reduced sum
  - Numerator
    - 29-gse
- Denominator
  - 30-gse

Figure A.37: Detailed view of the node *Associations*

Appendix B

Sample Tutorial Dialogues for Different Types of Tasks

The list of possible options for each prompt is given in italics. The correct one is the first option in the list and is given in bold.

The left-most path indicates what happens when a correct answer is given. The middle one is for incorrect answers. The right most is when a student indicates that he/she does not the answer.

B.1 Conceptual Database Design

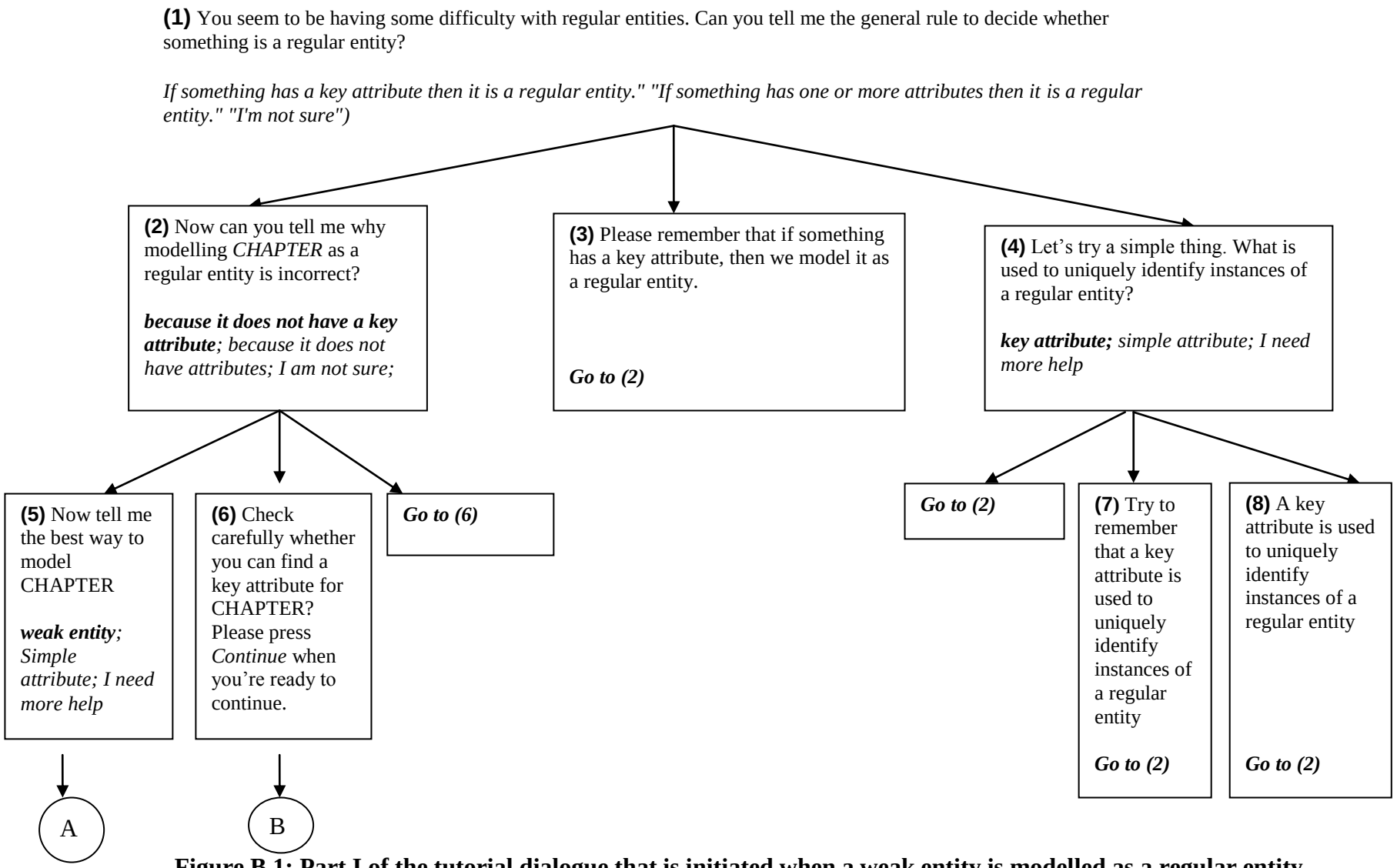


Figure B.1: Part I of the tutorial dialogue that is initiated when a weak entity is modelled as a regular entity

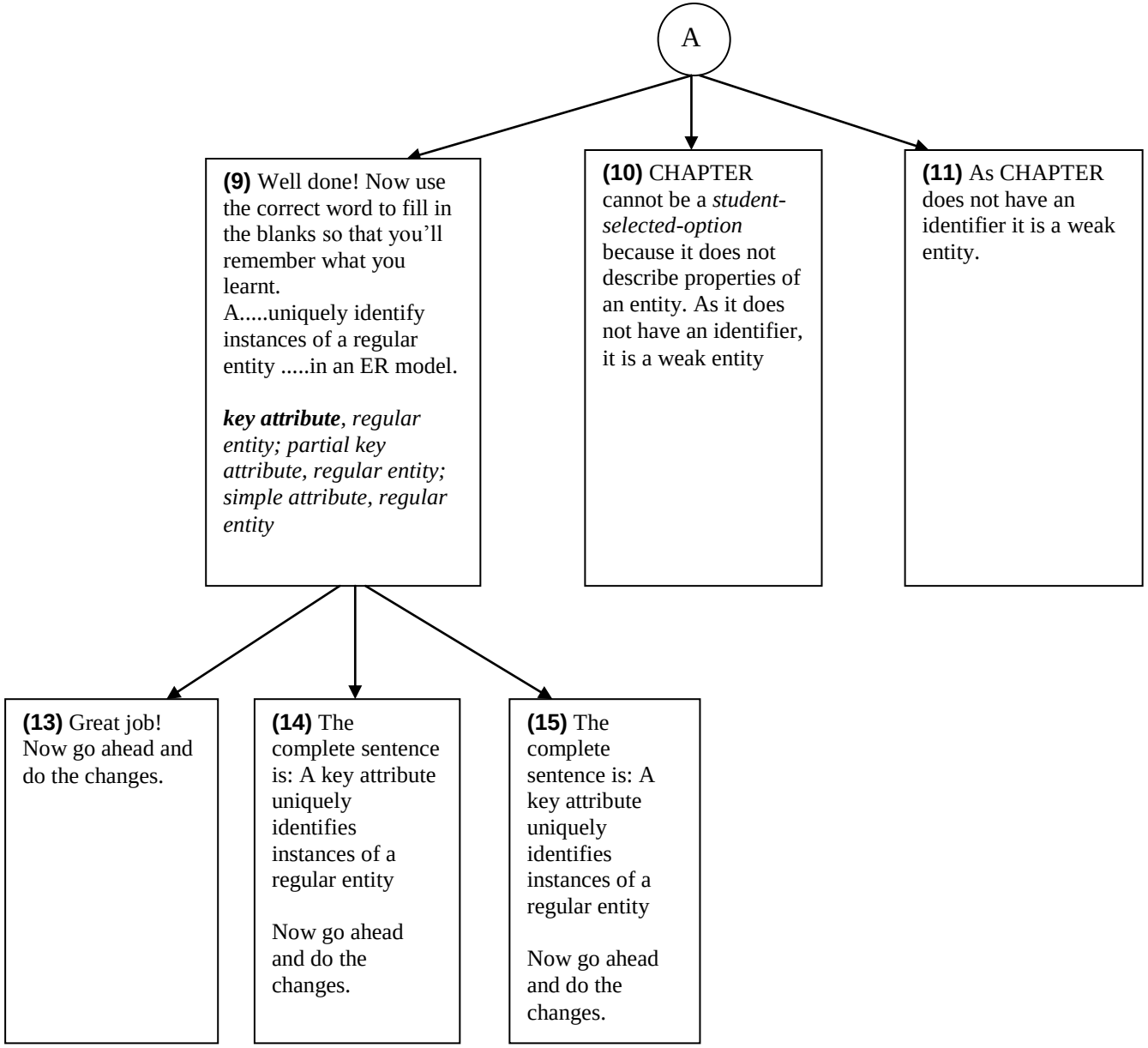


Figure B.2: Part II of the tutorial dialogue that is initiated when a weak entity is modeled on a regular entity

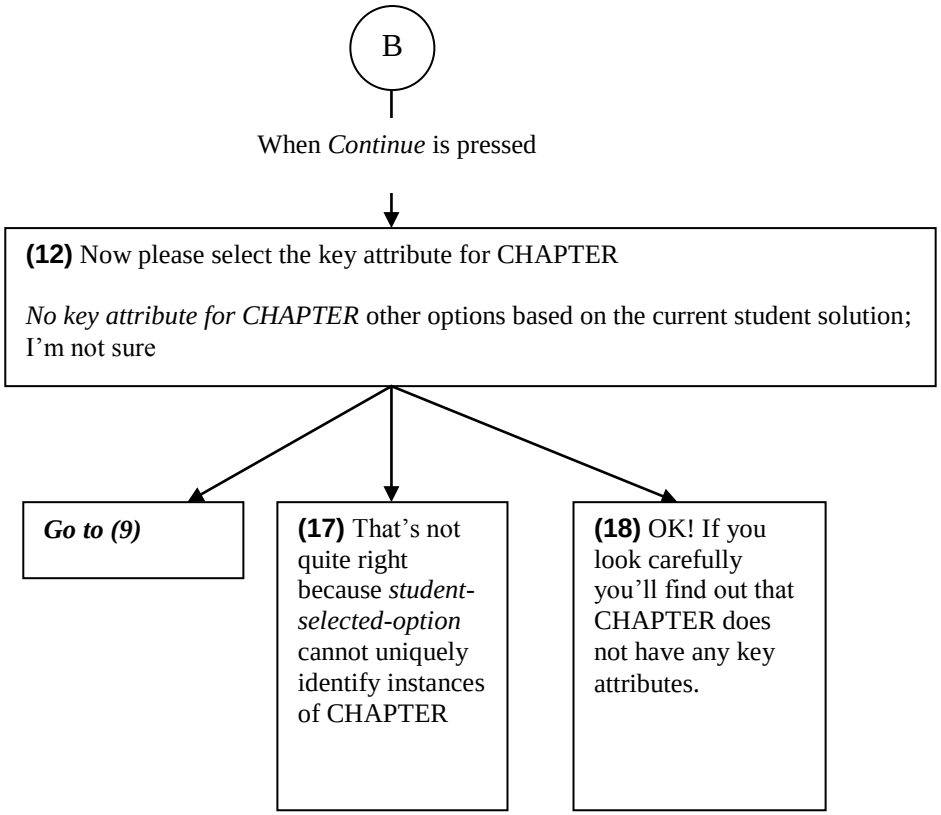


Figure B.3: Part III of the tutorial dialogue that is initiated when a weak entity is modeled on a regular entity

B.2 Data Normalization

(1) You seem to be having some difficulty with candidate keys. Let's look at candidate keys in detail. What is a candidate key?

The minimal set of attributes that determines all the attributes; all the attributes; I'm not sure

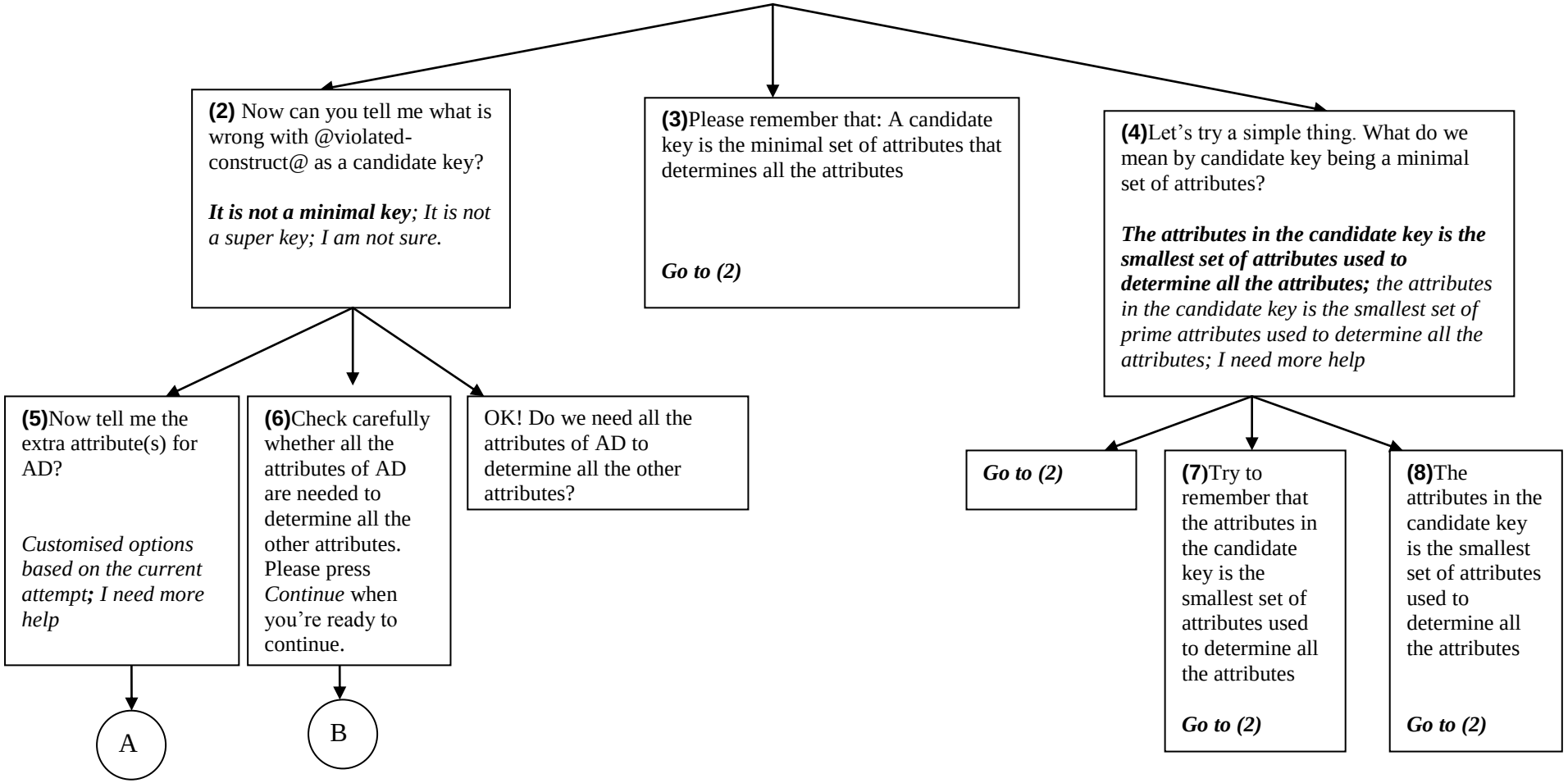


Figure B.4: Part I of the tutorial dialogue that is initiated when a non-minimal key is specified as a candidate key

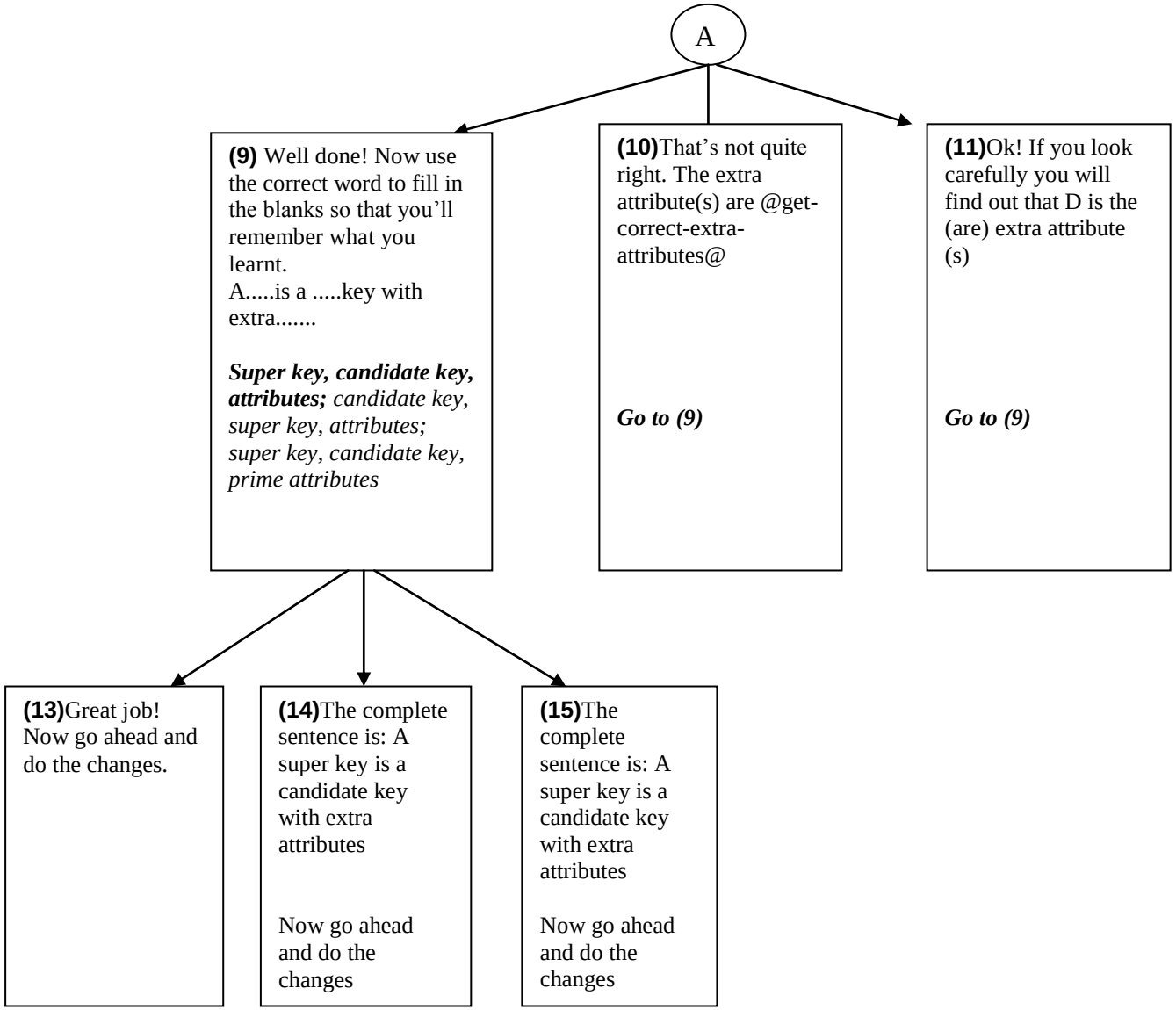


Figure B.5: Part II of the tutorial dialogue that is initiated when a non-minimal key is specified as a candidate key

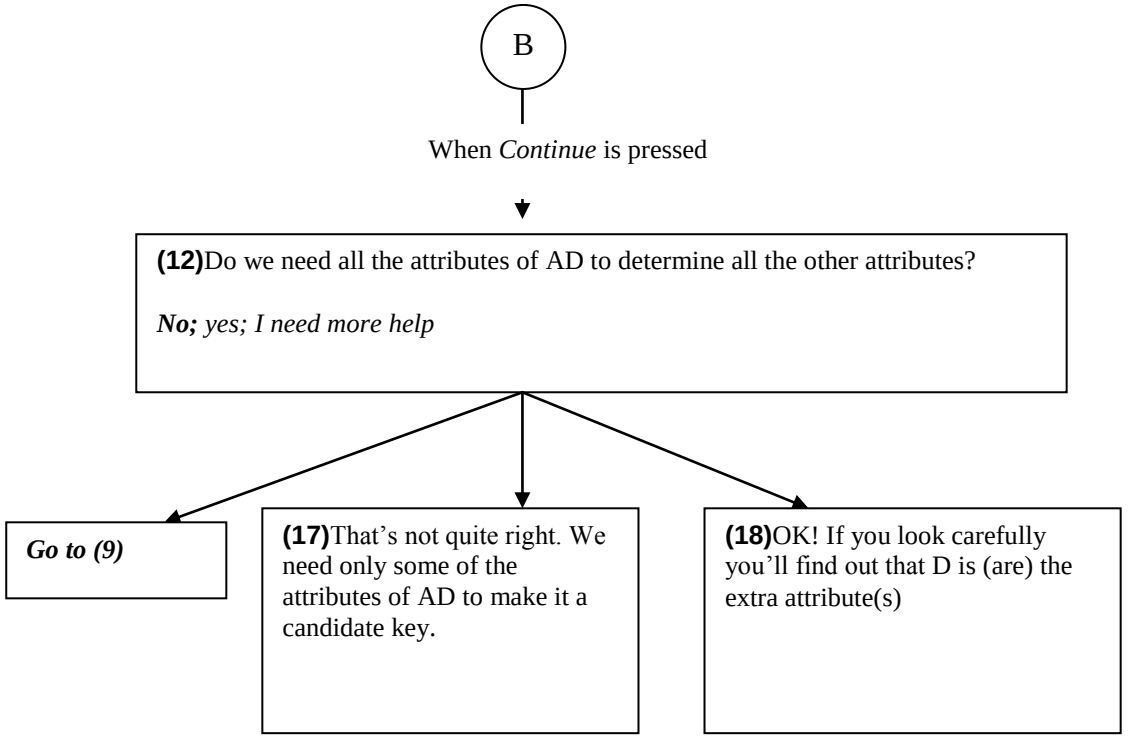


Figure B.6: Part III of the tutorial dialogue that is initiated when a non-minimal key is specified as a candidate key

B.3 Logical Database Design

(1) You seem to be having some difficulty with foreign keys. Let's look at foreign keys in detail. What is a foreign key?

Foreign key is a key that is used to establish the relationship between a weak entity and its owner; A foreign key is a key that is used to establish the relationship between two entities; I'm not sure

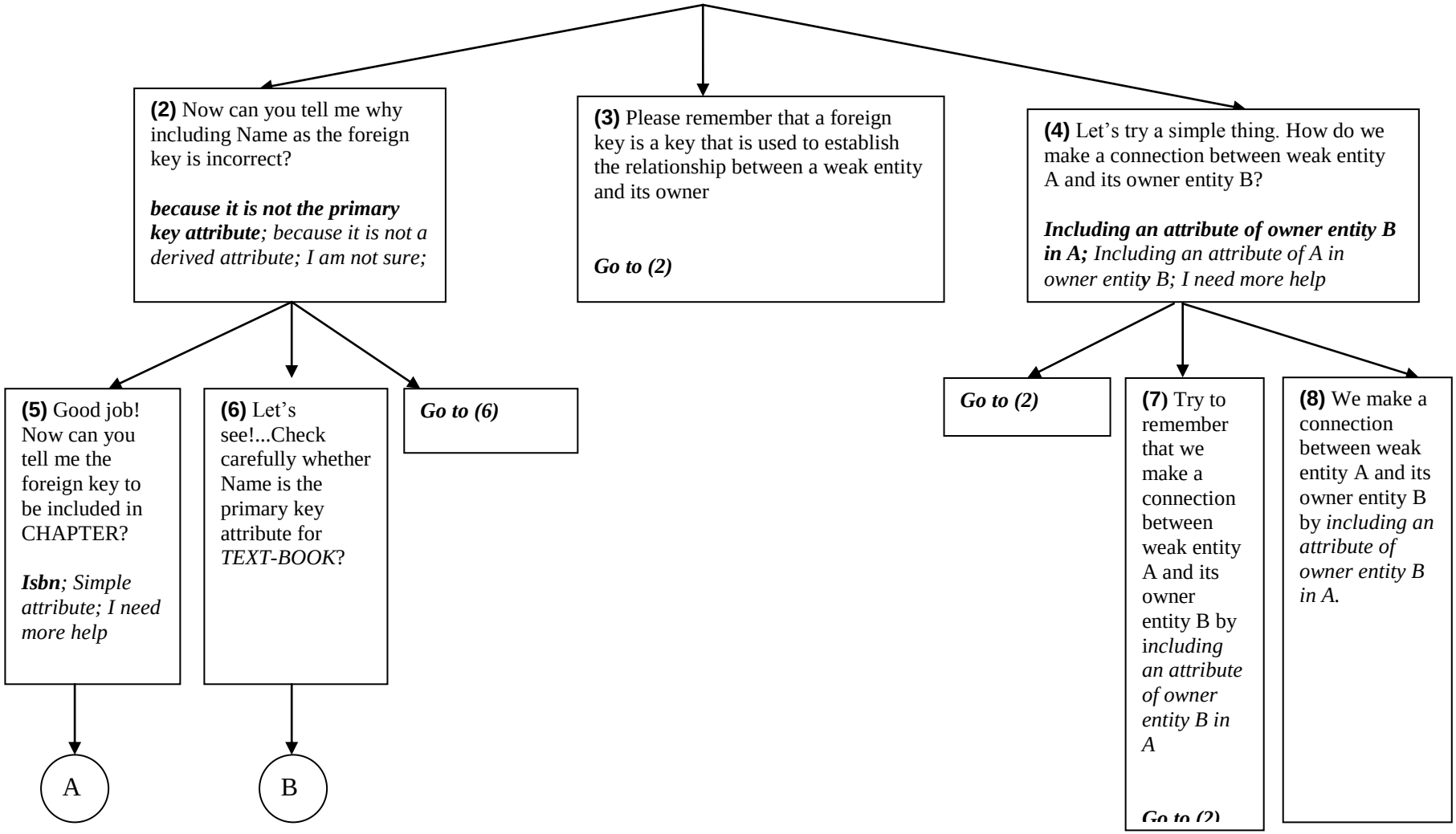


Figure B.7: Part I of the tutorial dialogue that is initiated when a non-primary key of an owner entity is specified as the foreign key

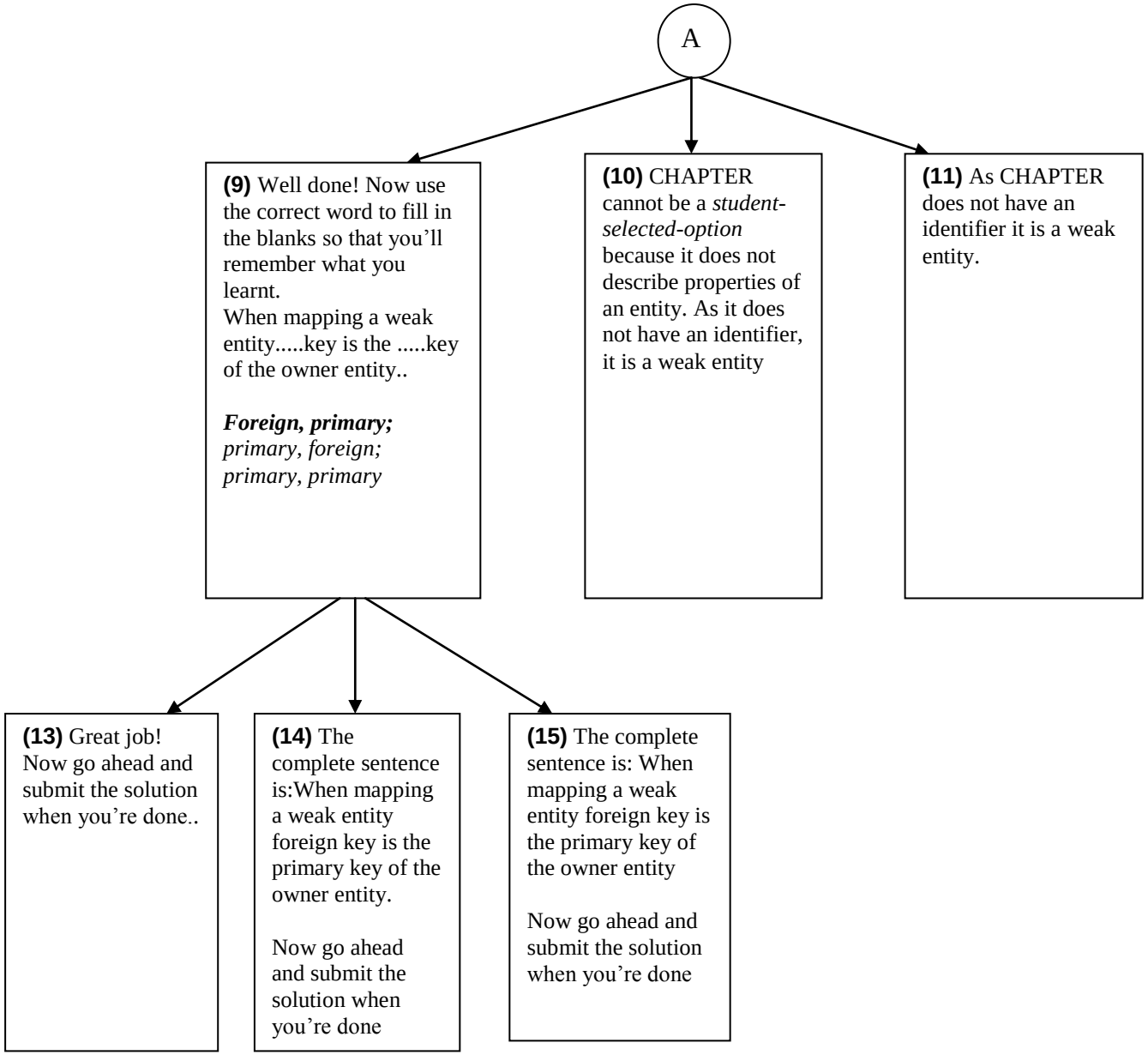


Figure B.8: Part II of the tutorial dialogue that is initiated when a non-primary key of an owner entity is specified as the foreign key

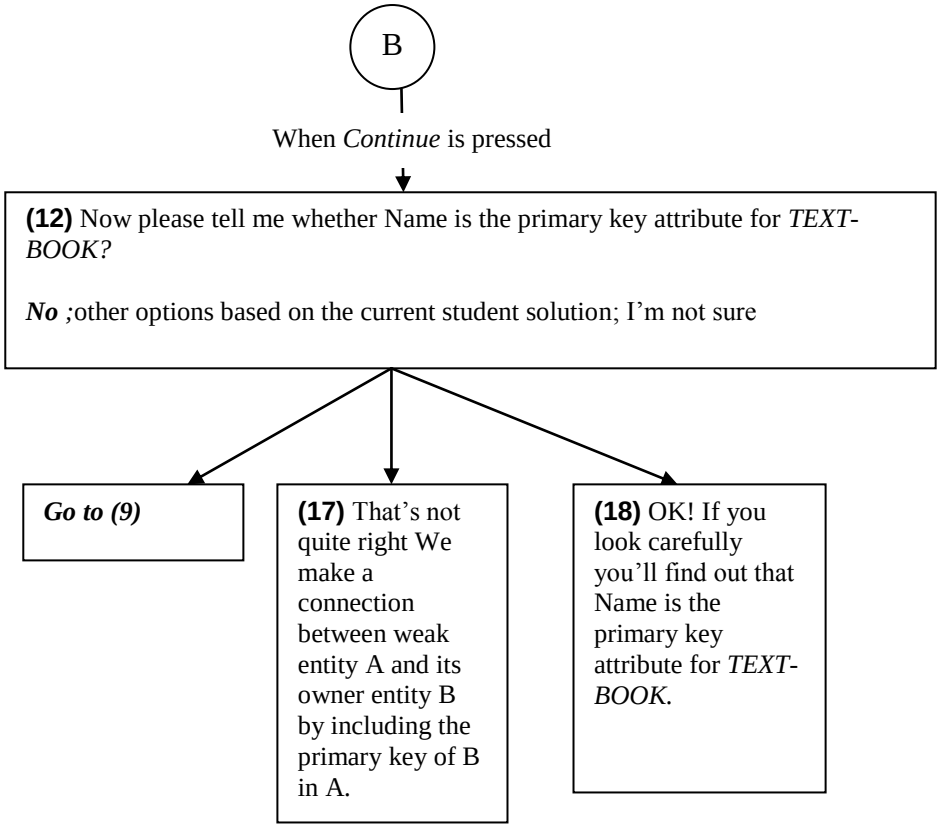


Figure B.9: Part III of the tutorial dialogue that is initiated when a non-primary key of an owner entity is specified as the foreign key

B.4 Fraction Addition

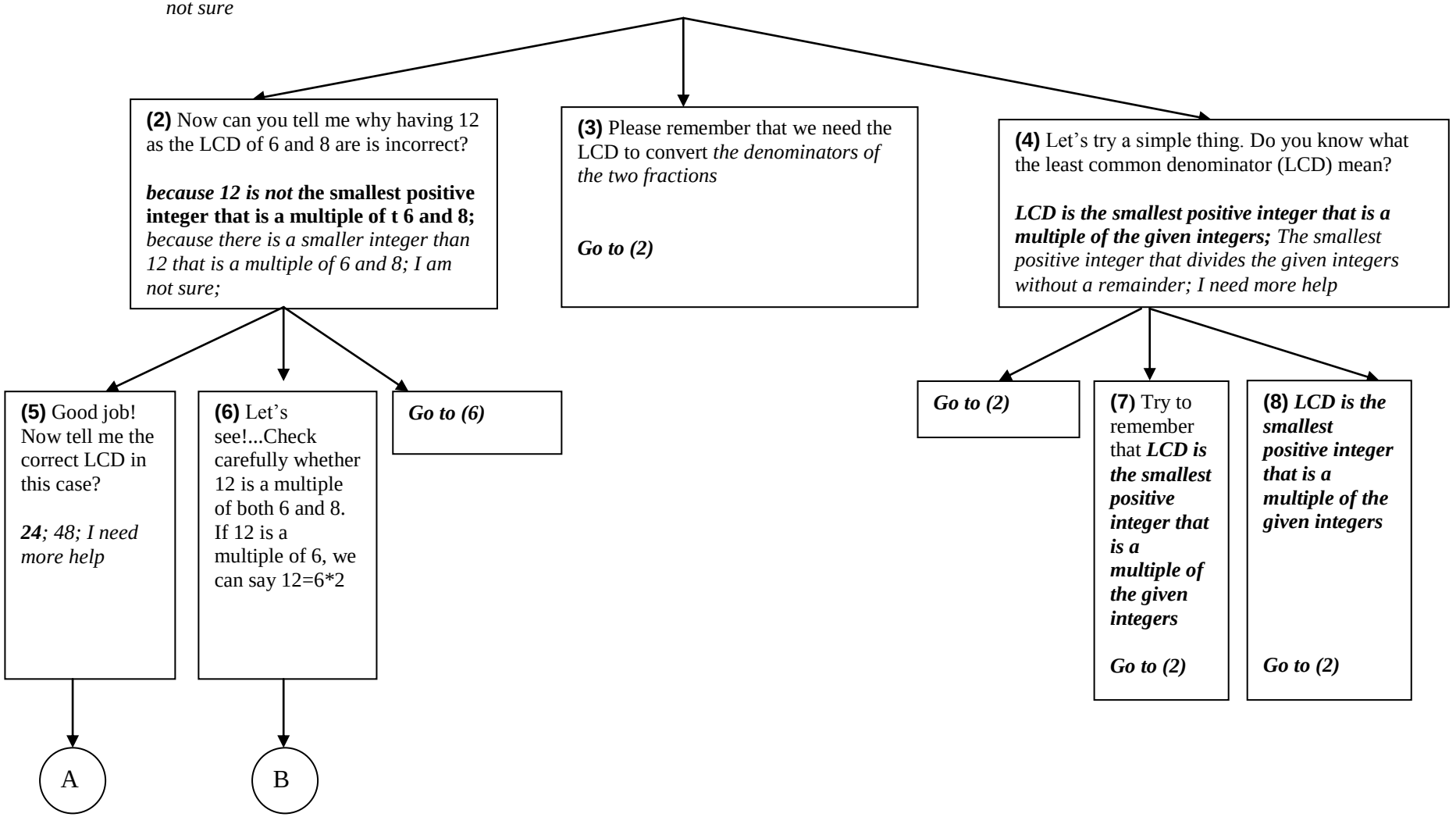


Figure B.10: Part I of the tutorial dialogue that is initiated when computed LCD for two given integers is incorrect

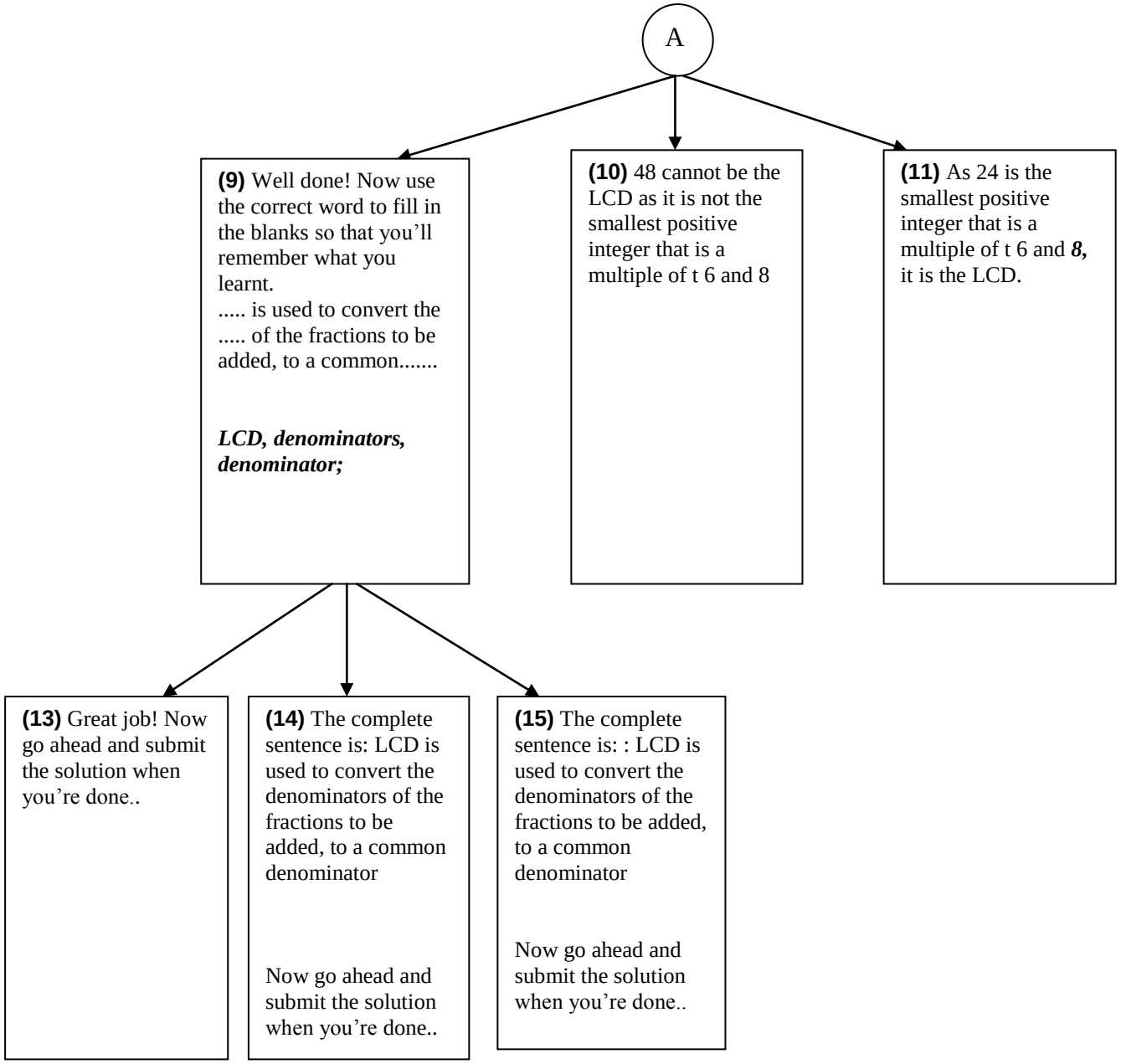


Figure B.11: Part II of the tutorial dialogue that is initiated when computed LCD for two given integers is incorrect

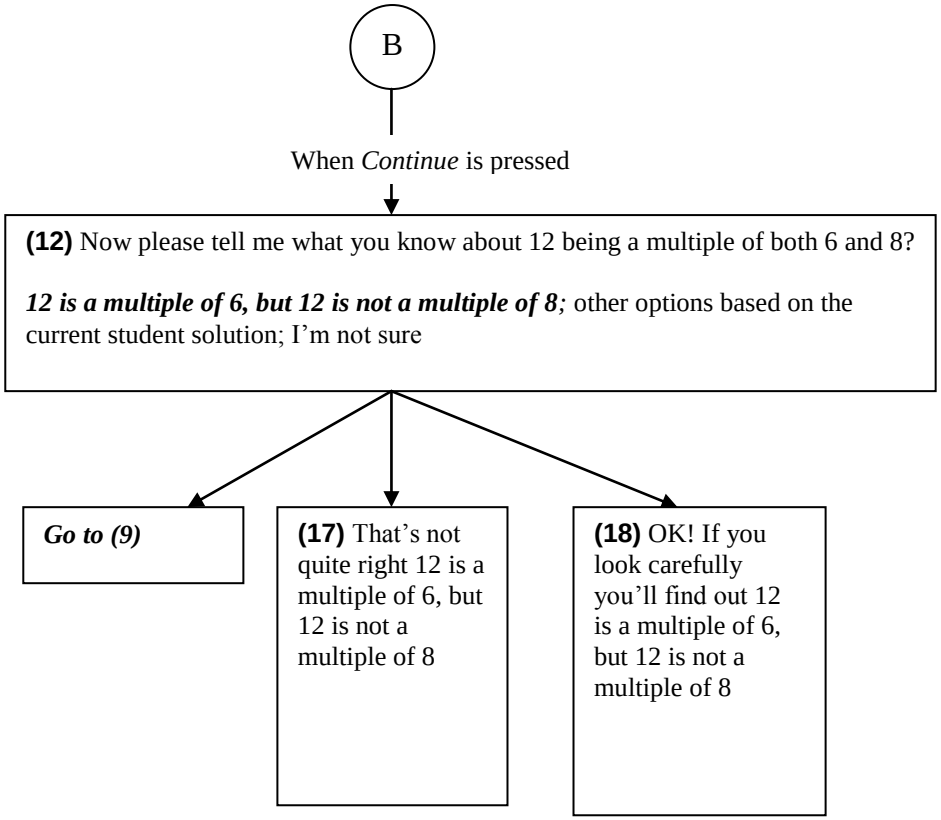


Figure B.12: Part III of the tutorial dialogue that is initiated when computed LCD for two given integers is incorrect

Appendix C

Forms for EER-Tutor Study

C.1 Letter of Approval from University of Canterbury

Human Ethics Committee
Tel: +64 3 364 2987 Extn 45588, Fax: + 64 364 2856
Email: human-ethics@canterbury.ac.nz

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Ref: HEC 2010/03/LR-PS

29 September 2010

Amali Weerasinghe
Department of Computer Science & Software Engineering
UNIVERSITY OF CANTERBURY

Dear Amali

Thank you for forwarding to the Human Ethics Committee a copy of the low risk application you have recently made for your research proposal "Facilitating tutorial dialogues in intelligent tutoring systems".

I am pleased to advise that this application has been reviewed and I confirm support of the Department's approval for this project.

With best wishes for your project.

Yours sincerely

Dr Michael Grimshaw
Chair, Human Ethics Committee

C.2 Pre-Test

Pre-test **User code:** _____

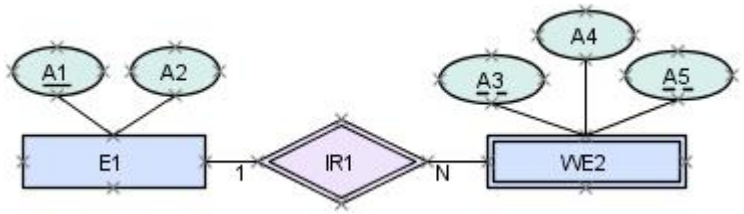
Question 1

Draw the diagram to model the following requirements.

> Each time a property is rented by a tenant we know the duration of the tenancy. Each tenant has unique customer number and a property has a unique property number.

Question 2

(a) Is the following diagram correct?
(b) Justify your answer.



Question 3

What are the possible values of a completeness constraint?

Question 4

Circle the correct answer.

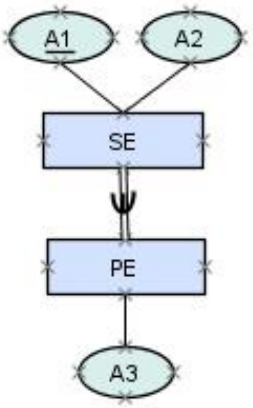
If an entity type has a multi-valued attribute, then

a. Each entity of this type can have one of several values for that attribute
b. There are some entities of this type that have more that one value for that attribute
c. Each entity of this type has more than one value for that attribute
d. There are many valid values for that attribute

Question 5

(a) Is the following diagram correct?
(b) Justify your answer.

.

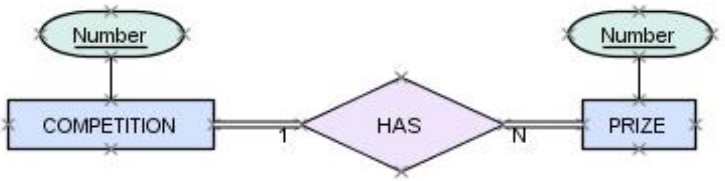


Question 6

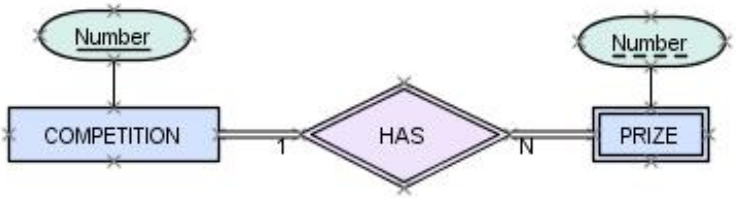
(a) Which of the given ER diagrams corresponds best to the following requirements?
(b) Justify your answer.

Each competition has a unique number. For every competition there is also a list of prizes identified by numbers unique only within a given list

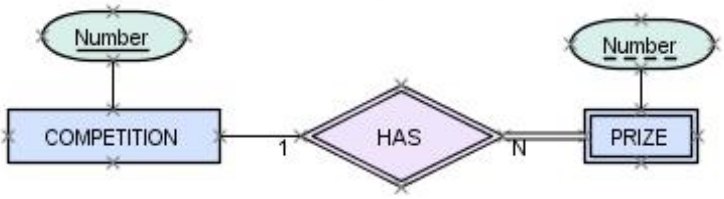
a



b



c



Question 7

How much interested are you in learning database design?

1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
Not interested at all Very interested

Question 8

How useful do you think learning database design is?

1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
Very useful Not useful at all

Question 9

How confident are you that you can learn database design using a computer tutor?
1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
Not confident at all Very confident

C.3 Post-Test

Post-test **User code:** _____

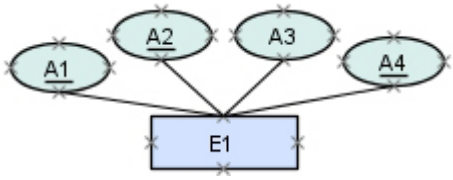
Question 1

Draw the diagram to model the following requirements.

Employees work on different projects. Each employee has a unique employee number and a project has a project code. For each project an employee is involved in the number of hours worked should be known.

Question 2

(a) Is the following diagram correct?
(b) Justify your answer.



Question 3

What is meant by a disjoint specialization?

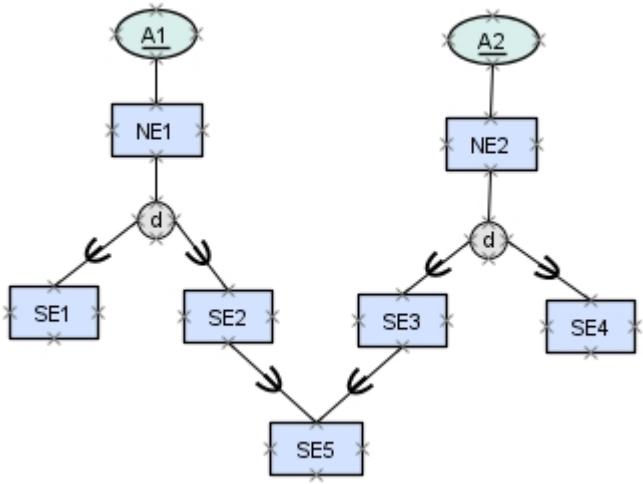
Question 4

A derived attribute is

a. An attribute whose values do not exist for every entity.
b. An attribute that has several components.
c. An attribute whose values are optional.
d. An attribute whose values can be derived from other attributes/relationships.
e. Don't know.

Question 5

(a) Is the following diagram correct?
(b) Justify your answer.

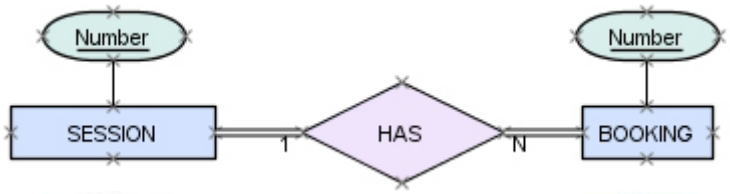


Question 6

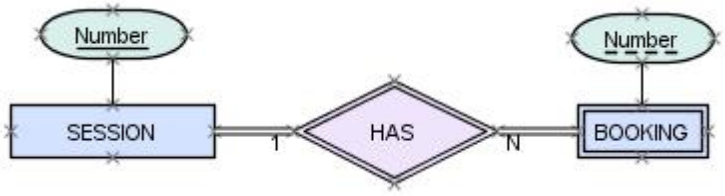
(a) Which of the given ER diagrams corresponds best to the following requirements?
(b) Justify your answer.

Sessions are identified by unique numbers. For some sessions, there might be lists of bookings identified by numbers unique only within a given session.

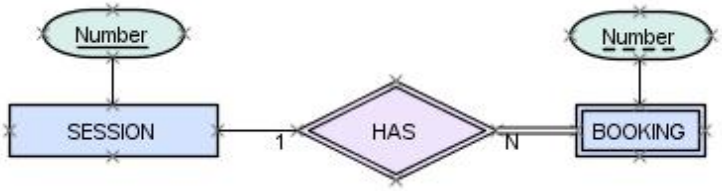
a



b



c



`

C.4 Questionnaire

Questionnaire –EER-Tutor User Code: ……..

1. How confident are you now that you can learn database design using a computer tutor?
   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Poor Excellent

2. How would you rate the overall quality of the dialogues of EER-Tutor?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Poor Excellent I haven't seen a single dialogue

3. How would you rate the length of the dialogues of EER-Tutor?
   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Too long Too short I haven't seen a single dialogue

4. How easy were the questions in the dialogues to understand?
   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Very Hard Very Easy I haven't seen a single dialogue

5. Were the dialogues useful in your learning (yes/no/not sure/haven't seen a single dialogue)?

_____

Please give us more details. _____
_____
_____

6. How did you feel about the system's prompts (i.e. prompts you received when you were inactive or abandoning problems) to give you help?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Too Frequent Too Infrequent I haven't seen a single prompt

7. What changes would you like to see in EER-Tutor's dialogues?
_____
_____
_____

8. If you have any other comments, please write them at the back of this sheet.

Appendix D

Forms for NORMIT Study

D.1 Letter of Approval from Victoria University of Wellington

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Phone 0-4-463 5676
Fax 0-4-463 5209
Email Allison.kirkman@vuw.ac.nz

MEMORANDUM

TO	Amali Weerasinghe
COPY TO	Pavle Mogin & Antonija Mitrovic
FROM	Dr Allison Kirkman, Convener, Human Ethics Committee

DATE	08 October 2010
PAGES	1

SUBJECT	**Ethics Approval: No 18071 Developing a General Model for Supporting Tutorial Dialogues in Intelligent Tutoring Systems**

Thank you for your applications for ethical approval, which have now been considered by the Standing Committee of the Human Ethics Committee.

Your applications have been approved from the above date and this approval continues until 1 January 2011. If your data collection is not completed by this date you should apply to the Human Ethics Committee for an extension to this approval.

Best wishes with the research.

Allison Kirkman
Convener

D.2 Pre-Test

Pre-Test **User Code: .....**

1. Relation R(A, B, C, D) is given, as well as a set of functional dependencies F = {C->D, C->A, B->C}. The primary key of R is:

   (i) A (ii) B (iii) C (iv) D (v) BC (vi) CD

Justify your answer.

---

2. For the same relation, identify the highest normal form it is in:

   (i) 1NF (ii) 2NF (iii) 3NF (iv) BCNF

Justify your answer.

---

3. Functional dependencies can be determined by examining the contents of tables in a database. (True/False)

---

4. A table is in BCNF if:
   a. the table is in 2NF
   b. the table has only one key
   c. there are no transitive functional dependencies
   d. every functional dependency has a superkey on its left-hand side
   e. the table is in 3NF
   f. there is a FD whose left-hand side is not a superkey

5. How much interested are you in learning data normalization?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5

   Not interested at all Very interested

6. How useful do you think learning data normalization is?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5

   Very useful Not useful at all

7. How confident are you that you can learn data normalization using a computer tutor?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5

   Not confident at all Very confident

D.3 Post-Test

Post-test **User Code:.....**

1. Relation R(A, B, C, D) is given, as well as a set of functional dependencies F = {A->B, BC->D, A->C}. The primary key of R is:

   (i) A (ii) B (iii) C (iv) D (v) AC (vi) BD

Justify your answer.

---

2. For the same relation, identify the highest normal form it is in:

   (i) 1NF (ii) 2NF (iii) 3NF (iv) BCNF

Justify your answer.

---

3. A relation can be in 3NF without being in 2NF. (True /False)

---

4. A functional dependency violates BCNF if it does not have:

   (i) a superkey on its right-hand side
   (ii) a key on its right-hand side
   (iii) a candidate key on its right-hand side
   (iv) a superkey on its left-hand side
   (v) a prime attribute on its right-hand side
   (vi) a prime attribute on its left-hand side

D.4 Questionnaire

Questionnaire –NORMIT User Code: ……..

1. How confident are you now that you can learn data normalization using a computer tutor?
   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Poor Excellent

2. How would you rate the overall quality of the dialogues of NORMIT?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Poor Excellent I haven't seen a single dialogue

3. How would you rate the length of the dialogues of NORMIT?
   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Too long Too short I haven't seen a single dialogue

4. How easy were the questions in the dialogues to understand?
   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Very Hard Very Easy I haven't seen a single dialogue

5. Were the dialogues useful in your learning (yes/no/not sure/haven't seen a single dialogue)?

_____

Please give us more details. _____
_____
_____

6. How did you feel about the system's prompts (i.e. prompts you received when you were inactive or abandoning problems) to give you help?

   1 ---------- 2 ---------- 3 ---------- 4 ---------- 5
   Too Frequent Too Infrequent I haven't seen a single prompt

7. What changes would you like to see in NORMIT's dialogues?
_____
_____
_____

8. If you have any other comments, please write them at the back of this sheet.

Appendix E

Publications & Co-authorship Forms

E.1 Published Papers

[1] Weerasinghe, A. and Mitrovic, A. (2006) Studying human tutors to facilitate self-explanation. Jhongli, Taiwan: 8th International Conference on Intelligent Tutoring Systems, 26-30 Jun 2006. In Lecture Notes in Computer Science (LNCS) 4053(Intelligent Tutoring Systems) 713-715. 10.1007/11774303_77.

[2] Weerasinghe, A. and Mitrovic, A. (2006) Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based Tutors. Jhongli, Taiwan: 8th International Conference on Intelligent Tutoring Systems, Workshop on Intelligent Tutoring Systems for Ill-Defined Domains, 27 Jun 2006. 56-64.

[3] Weerasinghe, A., Mitrovic, A. and Martin, B. (2007) Towards a General Model for Supporting Explanations to Enhance Learning. Los Angeles, CA, USA: 13th International Conference on Artificial Intelligence in Education (AIED 2007), 9-13 Jul 2007. In Frontiers in Artifical Intelligence and Applications 158 665-667.

[4] Weerasinghe, A., (2007) Developing a General Model for Supporting Self-Explanation for Intelligent Tutoring Systems Los Angeles, CA, USA: Young Researchers' Track of the 13th International Conference on Artificial Intelligence in Education (AIED 2007), 9-13 Jul 2007.

[5] Weerasinghe, A. and Mitrovic, A. (2008) A Preliminary Study of a General Model for Supporting Tutorial Dialogues. Taipei, Taiwan: 16th International Conference on Computers in Education (ICCE2008), 27-31 Nov 2008. 125-132.

[6] Weerasinghe, A., Mitrovic, A. and Martin, B. (2009) Towards Individualized Dialogue Support for Ill-Defined Domains. International Journal on Artificial Intelligence in Education 19(4): 357-379.

[7] Mitrovic, A. and Weerasinghe, A. (2009) Revisiting Ill-Definedness and the Consequences for ITSs. Brighton, UK: 14th International Conference on Artificial Intelligence in Education (AIED2009), 6-10 Jul 2009. In Frontiers in Artificial Intelligence and Applications 200, 375-382. http://dx.doi.org/10.3233/978-1-60750-028-5-375.

[8] Weerasinghe, A., Mitrovic, A., Van Zijl, M. and Martin, B. (2010) Evaluating the Effectiveness of Adaptive Tutorial Dialogues in database design. Putrajaya, Malaysia: 18th International Conference on Computers in Education ICCE 2010, 29 Nov-3 Dec 2010. In Proceedings of the 18th International Conference on Computers in Education 33-40.

[9] Weerasinghe, A., Mitrovic, A., Thomson, D., Mogin, P. and Martin, B. (2011) Evaluating a General Model of Adaptive Tutorial Dialogues. Auckland, New Zealand: 15th International Conference on Artificial Intelligence in Education (AIED 2011), 28 Jun-1 Jul 2011. In Lecture Notes in Computer Science (LNCS): Artificial Intelligence in Education 6738, 394-402

[10] Weerasinghe, A. and Mitrovic, A. (2011) Facilitating Adaptive Tutorial Dialogues in EER-Tutor. Auckland, New Zealand: 15th International Conference on Artificial Intelligence in Education (AIED 2011), 28 Jun-1 Jul 2011. In Lecture Notes in Artificial Intelligence 630-631.

Studying Human Tutors to Facilitate Self-Explanation

Amali WEERASINGHE and Antonija MITROVIC

Intelligent Computer Tutoring Group
Department of Computer Science and Software Engineering
University of Canterbury, Christchurch, New Zealand
{acw51,tanja}@cosc.canterbury.ac.nz

Abstract: This paper reports the first phase of a project with the goal of developing a general model of self-explanation support, which could be used in both open- and closed-ended domains. We studied how human tutors provide additional support to students learning with an existing intelligent tutoring system designed to help students learn database modelling. We report on the findings from this study, which will serve as the basis for defining the model. We also discuss directions for future work.

1. Introduction

Studies indicate that students acquire shallow knowledge even in the most effective Intelligent Tutoring Systems (ITS) [1]. Self-explanation was shown to facilitate the acquisition of deep knowledge [2]. Several ITSs were enhanced with self-explanation support in domains such as physics [3], mathematics [1], database design [6] and data normalization [5]. With the exception of database design, all these domains are closed-ended, as problem solving is well structured, and therefore self-explanation expected from learners can be clearly defined. Database design is an open-ended task: the final result can be defined in abstract terms, but there is no algorithm to find it. Although the above ITSs were shown to improve student performance, none of these self-explanation models have been used in both open- and closed-ended domains.

Our long-term goal is to develop a model to facilitate self-explanation which can be used in both open- and closed-ended domains. We have chosen Entity-Relationship (ER) modelling as the open-ended domain, and ER-to-relational mapping as a closed-ended domain. The later task is a well-formed one, and therefore is a deterministic algorithm that students learn in database courses. EER-Tutor [7] and ERM-Tutor [4] are two existing constraint-based tutors. Our goal is to develop a general self-explanation model that can be used to enhance these systems.

In order to develop a model for self-explanation, we need to consider three basic decisions: when to prompt for self-explanation, what to self-explain and how to obtain self-explanation from learners. As the first step, we conducted a study to observe how students interacted with the EER-Tutor, while providing additional help by a human tutor through a chat interface. Section 2 presents this study. The next section discusses the findings of this study and how they can be incorporated in a self-explanation model. Section 3details the conclusions and the directions of future work section.

2. Preliminary Study

The study was conducted in August 2005 at the University of Canterbury, and involved volunteers enrolled in an introductory database course and professional tutors. The professional tutors will be referred to as tutors, while EER-Tutor as the system hereafter. EER-Tutor provides a problem-solving environment and complements classroom instruction. The version of EER-Tutor used in the study was enhanced with a chat interface, so that the tutors could provide one-to-one feedback to students. We wanted to make the bandwidth between the student and the tutor very similar to that between the student and the ITS. As a result, tutors could observe only the students' interactions with the ITS. Participants interacted with the system in one room and the tutors observed their interactions in another room.

The tutors were not given any specific instructions on providing assistance to students. Student participants were not told that a human tutor was involved in the study. Students also could ask for help through the chat interface or the *More Help* button in the interface. All interactions were recorded. Students themselves decided when to end the session. All participants filled out a questionnaire on their perceptions about the system and interventions through the chat interface. The tutors were also interviewed to understand their views on the tutoring experience.

3. Observations and Prototype for the Self-Explanation Model

Seven students and four professional tutors participated in the study, with at most two students per tutor. The average duration of sessions was 85 minutes (sd=20). The average number of problems attempted was 11 (sd = 5), and all the participants completed all the problems attempted. The timing of tutor interventions differed significantly. Some tutors intervened in the first problem in which the student needed help, while in other sessions, the tutors intervened mostly in 4^{th} or the 5^{th} problem. In one situation, the tutor waited until the 19^{th} problem to intervene.

The self-explanation model will be developed on the basis of the findings from this study. The model will decide when and what to self-explain, and how to obtain self-explanations. As all tutors provided delayed feedback, which was well-received by the participants, the model will provide delayed feedback. With delayed feedback, specific guidelines to decide on the timing of interventions need to be incorporated into the model. In the study, delayed feedback was provided in the following situations: (i) the student has been inactive or moving the mouse aimlessly for a pre-determined period of a time, (ii) the student has made the same mistake repeatedly or (iii) the student seems to be reacting to feedback without much reflection.

In the first scenario, it will be beneficial to prompt the student to ask a question in order to understand the difficulty in completing the solution, to which the system can respond appropriately. This either requires natural language capabilities or obtaining the response through menu options. For instance, we can ask the student which concept he/she is having difficulties with, and provide a menu for the student to select the concept he/she needs assistance with. As noted in (ii), if the student makes the same mistake repeatedly, it is clear that there is a misconception or gap in his/her knowledge. In such a situation, it will be more beneficial to provide a problem-independent explanation initially. Then the student may need assistance to understand how to apply the domain concept to the current state of the problem. A student seems

to be reacting to feedback without reflection if he/she makes a single change without reflecting on the other changes that need to be performed as a result. In such situations, the student will be prompted to reflect on other related changes.

The self-explanation model also needs to decide how to prompt learners to self-explain. As explained earlier, we have seen that human tutors provide problem-independent explanations when there is evidence that a student has difficulty with a domain concept. Later on, the student can be prompted to understand how the corresponding domain concept relates to the current problem state. At other times, the student may have difficulty with the current problem. In such a situation, the student can be guided using a series of prompts ranging from rephrasing feedback, discussing the problem-specific details to providing the answer directly. If the system has natural language processing capabilities, students would be able to specify partial answers and correspond with the ITS in a natural manner.

4. Conclusions and Future Work

This research focuses on developing a self-explanation model for both open- and closed-ended domains. As the first step, we conducted a study to observe how tutors help students to solve a problem using the EER-Tutor. In addition to the system's feedback, the students were prompted by human tutors through a chat interface. Although the tutors used different kinds of prompts, all of them provided delayed feedback and guided the students towards the solution without giving the answer directly. According to the questionnaire responses, both timing and content of interventions were well received by the students. They also felt that the help received through the chat interface was very useful for understating mistakes on their own, providing an opportunity for self-explanation and reflection.

The findings from the study are being used to develop the model of self-explanation, which will be used in the next study with ERM-Tutor to understand its applicability in a closed-ended domain. If necessary, the model will be modified and implemented in both EER-Tutor and ERM-Tutor, which will later be evaluated in authentic class room environments.

References

1. Aleven, V., Koedinger, K. R., Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Lajoie, S.P. and Vivet, M.(eds.), AIED 1999, IOS Press, 199-206.
2. Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. Advances in Instructional Psychology, (2000) 161-238.
3. Conati, C., VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. Artificial Intelligence in Education, 11 (2000) 389-415.
4. Milik, N., Marshall, M., Mitrovic, A. Teaching Logical Database Design in ERM-Tutor. Accepted for publication in ITS2006.
5. Mitrovic, A. The Effect of Explaining on Learning: a Case Study with a Data Normalization Tutor. In: C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (eds) Proc. AIED 2005, 499-506.
6. Weerasinghe, A., Mitrovic, A. Supporting Self-Explanation in an Open-ended Domain. In: Gh.Negoita, M., Howlett, R. J., Jain, L.C. (eds.) Proc. of KES 2004 306-313.
7. Zakharov, K., Mitrovic, A., Ohlsson, S., Feedback Micro-engineering in EER-Tutor. In: Looi, C-K., McCalla, G., Bredeweg, B., Breuker, J. (eds.) Proc. AIED 2005, (2005) 718-725.

Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based Tutors

Amali Weerasinghe
Intelligent Computer Tutoring Group
Dept. of Computer Science & Software Engineering, University of Canterbury, New Zealand
acw51@cosc.canterbury.ac.nz

Antonija Mitrovic
Intelligent Computer Tutoring Group
Dept. of Computer Science & Software Engineering, University of Canterbury, New Zealand.
tanja@cosc.canterbury.ac.nz

Abstract. We present the first phase of a project with the goal of developing a general model of self-explanation support, which could be used in constraint-based tutors for both well- and ill-defined domains. We studied how human tutors provide additional support to students learning with an existing intelligent tutoring system designed to help students learn an ill-defined task (database modeling using the ER model). Although the tutors were not given specific instructions to facilitate self-explanation, there were instances when self-explanation support was provided. Analysis of these interactions indicates that they have helped the students to improve their understanding of database design. These findings will serve as the basis for defining the self-explanation model. We also discuss directions for future work.

Keywords: self-explanation, intelligent tutoring systems, student modeling, ill-defined tasks

INTRODUCTION

Studies indicate that some students acquire shallow knowledge even in the most effective Intelligent Tutoring Systems (ITS) (Aleven et al., 1999). Self-explanation (SE) is described as an "*activity of explaining to oneself in an attempt to make sense of new information, either presented in a text or in some other medium*" (Chi, 2000), and has been shown to facilitate the acquisition of deep knowledge (Chi et al., 1989). There are several ITSs that facilitate self-explanation, most of them teaching well-defined tasks. For example, SE-Coach (Conati, and VanLehn, 2000) prompts students to explain solved physics examples. In the PACT Geometry Tutor (Aleven et al., 1999), students explain solution steps by selecting definitions and theorems from a glossary. NORMIT (Mitrovic et al., 2004) is an ITS for data normalization, in which students are expected to self-explain while solving problems. All these domains are closed-ended, as problem solving is well structured, and therefore self-explanation expected from learners can be clearly defined. Database design is an open-ended task: the final result can be defined in abstract terms, but there is no algorithm to find it. Constraint-based tutors have demonstrated their effectiveness in teaching ill-defined tasks, such as database design and querying (Mitrovic et al., 2004), and software design (Baghaei et al., 2005). We have extended the database design tutor with a SE facility (Weerasinghe and Mitrovic 2006). Even though all the above mentioned ITSs facilitate self-explanation, only SE-Coach supports adaptive self-explanation customised to the student's knowledge and self-explanation skills.

Our long-term goal is to develop a model of self-explanation which will provide adaptive support to learners for both well- and ill-defined tasks. Since we previously implemented self-explanation support for the database design tutor, the initial work on this project started with the same tutor. We are currently developing an SE model, which will be incorporated into EER-Tutor (Zakharov et al., 2005). In order to develop this model, we need to consider three basic decisions: when to prompt for self-explanation, what to self-explain and how to obtain self-explanations from learners. As the first step, we conducted an observational study to investigate how students interacted with EER-Tutor, while getting additional help by a human tutor through a chat interface.

A brief discussion of database design is given in the following section. We then discuss the functionality of EER-Tutor, followed by a description of the observational study. Analysis of the student interactions are presented in the Observations Section. We then discuss how the findings from the study can be incorporated in the self-explanation model. Future work and conclusions are presented in the final section.

DATABASE DESIGN

Database design is a process of generating a description of a database using a specific data model. Most database courses teach conceptual database design using the Entity-Relationship (ER) model, a high-level data model originally proposed by Chen (1976). The ER model views the world as consisting of *entities*, and *relationships*

between them. The entities may be physical or abstract objects, roles played by people, events, or anything else data should be stored about. Entities are described in terms of their important features (*attributes),* while relationships represent various associations between entities, and also may have attributes. There is no algorithm to use to derive the ER schema from a given set of requirements. The learner needs to decide on the appropriate constructs to use, such as types of attributes/entities. For example, the learner might be given a problem illustrated in Figure 1 (note that this is a very simple problem). From the problem text, it is obvious that *students* and *groups* are of importance. Therefore, the learner might start by drawing the entities first. Each student has an id, and the learner needs to use his/her world knowledge to realize that ids are unique, and therefore represent that attribute as a key attribute (shown on the diagram as underlined). The number assigned to each group is unique, and therefore it should also be a key attribute. In Figure 1, the student has made a mistake by showing GROUP as a weak entity, and group number as a partial key. Next, the learner has to think about the relationships between identified entities. In the problem shown in Figure 1, students work in groups, and for each possible association between a student and a group, it is necessary to represent the role. The Role attribute describes the association, and therefore it should be an attribute of the relationship. The student also needs to specify other integrities, such as cardinality ratios (shown as N on the diagram) and participations (shown as single or double lines).

As can be seen from this simple case, there are many things that the student has to know and think about when designing databases. The student must understand the data model used, including both the basic building blocks available and the integrity constraints specified on them. In real situations, the text of the problem would be much longer, often ambiguous and incomplete. To identify the integrities, the student must be able to reason about the requirements and use his/her own world knowledge to make valid assumptions.

Database design, similar to other design tasks, is an ill-defined task, because the start/goal states and the problem-solving algorithm are underspecified (Reitman, 1964). The start state is usually described in terms of ambiguous and incomplete specifications. The problem spaces are typically huge, and operators for changing states do not exist. The goal state is also not clearly stated, but is rather described in abstract terms. There is no definite test to decide whether the goal has been attained, and consequently, there is no best solution, but rather a family of solutions. Design tasks typically involve huge domain expertise, and large, highly structured solutions.

Although design tasks are underspecified, Goel and Pirolli (1992) identify a set of 12 invariant features of design problem spaces, such as problem structuring, distinct problem-solving phases, modularity, incremental development, control structure, use of artificial symbol systems and others. Problem structuring is the necessary first phase in design, as the given specifications of a problem are incomplete. Therefore, the designer needs to use additional information that comes from external sources, the designer's experience and existing knowledge, or needs to be deduced from the given specifications. Only when the problem space has been constructed via problem structuring, problem solving can commence. The second feature specifies three problem-solving phases: preliminary design, refinement and detail design. Design problem spaces are modular, and designers typically decompose the solution into a large number of sparsely connected modules and develop solutions incrementally. When developing a solution, designers use the limited-commitment mode strategy, which allows one to put any module on hold while working on other modules, and return to them at a later time.

In previous work, we have shown that constraint-based tutors are highly effective in teaching ill-defined tasks such as database design (Suraweera & Mitrovic, 2004) and query definition (Mitrovic & Ohlsson, 1999; Mitrovic et al., 2004). Our tutors compare the student's solution to a pre-specified ideal solution, which captures the semantics of the problem, thus eliminating the need for a problem-solver, which is difficult (or even impossible) to develop for such instructional domains. The constraint-based tutors are capable of identifying alternate correct solutions as constraints check that the student's solution contains all the necessary elements, even though it might be different from the ideal solution specified by the teacher. Goel and Pirolli (1988) argue that design problems by their very nature are not amenable to rule-based solutions. On the other hand, constraints are extremely suitable for representing design solutions: they are declarative, non-directional, and can describe partial or incomplete solutions. A constraint set specifies all conditions that have to be simultaneously satisfied without restricting how they are satisfied. Each constraint tests a particular aspect of the solution, and therefore supports modularity. Incremental development is supported by being able to request feedback on a solution at any time. At the same time, CBM supports the control structure used by the designer (student), as it analyses the current solution looking at many of its aspects in parallel: if a particular part of the solution is incomplete, the student will get feedback about missing constructs. CBM can be used to support all problem-solving phases. Therefore, we believe that CBM can be applied to all design tasks.

EER-TUTOR: ENHANCED ENTITY RELATIONSHIP TUTOR

EER-Tutor is aimed at the university-level students learning conceptual database design. For a detailed discussion of the system, see (Zakharov et al., 2005); here we present some of its basic features. The system complements traditional instruction, and assumes that students are familiar with the ER model. The system consists of an interface, a pedagogical module, which determines the timing and content of pedagogical actions,

and a student modeller, which analyses student answers and generates student models. EER-Tutor contains a set of problems and the ideal solutions to them, but has no problem solver. In order to check the student's solution, EER-Tutor compares it to the correct solution, using domain knowledge represented in the form of more than 200 constraints. It uses Constraint-Based Modelling (Mitrovic, et al, 2004) to model the domain and student's knowledge. The interface (illustrated in Figure 1) is composed of three windows tiled horizontally. The top window displays the current problem and provides controls for stepping between problems, submitting a solution and selecting feedback level. The middle window is the main working area, in which students draw ER diagrams.

Feedback from the system is grouped into six levels according to the amount of detail: *Correct*, *Error Flag*, *Hint*, *Detailed Hint*, *All Errors* and *Solution*. The first level of feedback, *Correct*, simply indicates whether the submitted solution is correct or incorrect. The *Error Flag* indicates the type of construct (e.g. entity, relationship) that contains the error. For example, when the solution in Figure 1 is submitted, *Error Flag* provides the message *Check your entities, that's where you have some problems*. This is associated with the error GROUP being modelled as a weak entity instead of a regular entity. *Hint* and *Detailed Hint* offer a feedback message generated from the first violated constraint. For the solution in Figure 1, the hint message is *Check whether all the weak entities are necessary. Check whether some of your weak entities should be represented using some other type of construct*. On the other hand, the corresponding detailed hint is more specific: GROUP *should not be an entity. It may be extra or you may want to represent it using some other type of construct*, where the details of the erroneous object are given. Not all detailed hint messages give the details of the construct in question, since giving details on missing constructs would give away solutions. A list of feedback messages on all violated constraints is displayed at the all errors level (as indicated in the right-hand pane in Figure 1). The ER schema of the complete solution is displayed at the final level (solution level).

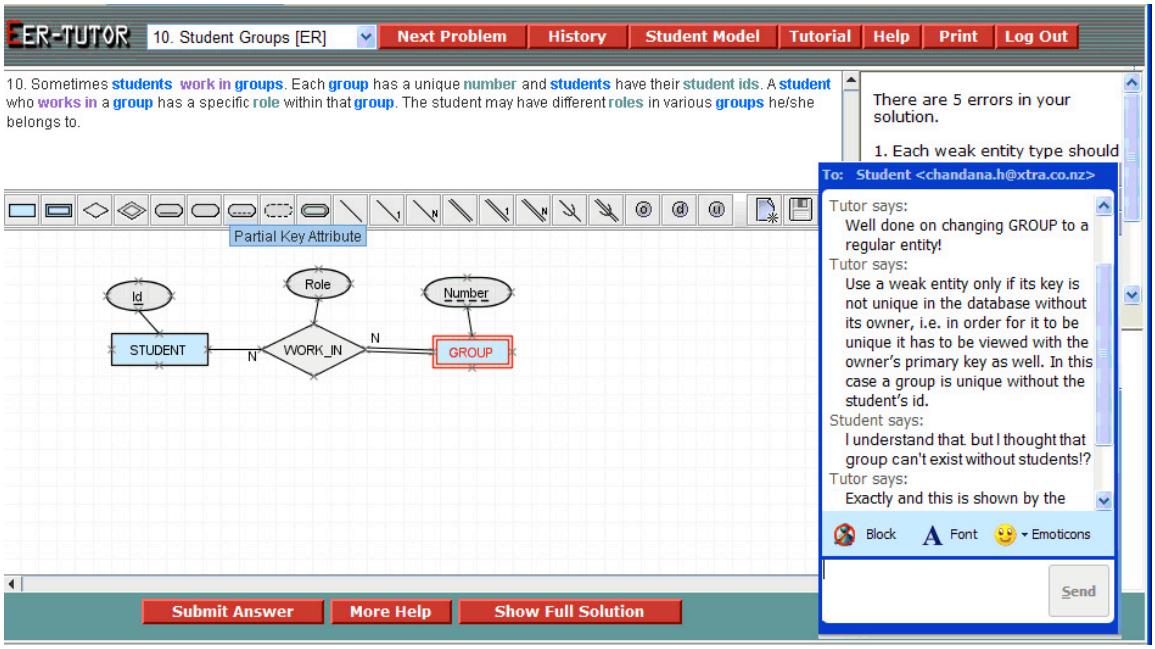


Fig. 1. Interface of the enhanced EER-Tutor

When the student submits the first attempt at a problem, a simple message indicating whether or not the solution is correct is given. The level of feedback is incremented with each submission until the feedback level reaches the detailed hint level. In other words, if the student submits the solutions four times, the feedback level would reach the detailed hint level, thus incrementally providing more detailed messages. Automatically incrementing the levels of feedback is terminated at the detailed hint level to encourage the student to concentrate on one error at a time rather than all the errors in the solution. The system also gives the student the freedom to manually select any level of feedback according to their needs.

PRELIMINARY STUDY

The study was conducted in August 2005 at the University of Canterbury, and involved students enrolled in an introductory database course and experienced tutors. These experienced tutors will be referred to as tutors, while EER-Tutor as the system or the ITS hereinafter. All the tutors had several years of experience providing assistance to students in labs and/or teaching small groups. The study was scheduled only after the relevant material was taught in the classroom. The version of EER-Tutor used was enhanced with a chat interface (Figure 1), so that the tutors could provide one-to-one feedback to students. We wanted to make the bandwidth between

the student and the tutor to be similar to that between the student and the ITS. As a result, tutors could observe only the students' interactions with the ITS. Participants and tutors were located in separate rooms.

The tutors were expected to guide the students towards solutions using appropriate methods like asking questions etc. However, they were not given any specific instructions on providing assistance. Student participants were not told that a human was involved in the study. They also had the opportunity to initiate intervention through the chat interface or the *More Help* button in the interface.

At the beginning of the study, the students were asked to sit a pre-test online. All learner interactions were recorded. Students were expected to use the system for at least an hour. However, the students themselves decided when to end the session. Although initially we wanted the participants to sit a post-test immediately after the study, it was not possible due to another evaluation study which was conducted simultaneously. Therefore, the post-test was administered later on. All participants were asked to fill out a questionnaire at the end of the session to understand their perceptions about the system and interventions through the chat interface. At the end of each session, the tutors were also interviewed to understand their views on the tutoring experience.

We initially analysed the recordings without the tutors, to investigate how students were prompted by different tutors. As the second step, whenever possible, the recordings were analysed with the tutors to clarify how they decided on the timing and the level of feedback provided through the chat interface.

The experimental set up of this study varies from previous studies of tutorial dialogue in a number of ways. First, the tutors were expected to provide additional support to the feedback given by the system. The tutors were also expected to respond to learners' questions. This contrasts with those studies of Chi. et al. (Chi and Siler, 2001) and Graesser, Person et al. (1995) in which the tutor was expected to lead the dialogue through a series of questions. Second, the learner interacts both with the system and the tutor. Although Merill et al. (1992) have studied tutorial dialogues in the context of problem-solving, the tutor was the only source of feedback for the student as s/he solved problems on paper. Finally, the tutors in our study needed to decide not only how to guide the student but also when. This differs significantly from the study in which the tutors analysed recorded interactions of students to perform motivation diagnosis (De Vicente and Pain, 2002).

OBSERVATIONS

Seven students and four professional tutors participated in the study, with at most two students per tutor. The mean on the pretest was 75.5% (sd=17.9), which was higher than the performance of the whole class (mean=58.1, sd=23.5). We expected this, as the participants were self-selected. Still the range of background knowledge was sufficiently large (ranging from 57% to 100%). Only two students have completed the post-test, hence it is not possible to compare the effect the learner interactions had on performance. The average duration of the sessions was 85 minutes (sd=20). The average number of attempted problems was 11 (sd = 5), and all participants completed all attempted problems. Average number of attempts per problem is 2.8 (i.e. received feedback from EER-Tutor that many times). We discuss observations in two different categories: (i) type of feedback provided in the interventions and (ii) timing of interventions.

Type of Feedback Provided

The interactions between the tutors and the students were analysed to identify different episodes, each pertaining to a single topic (Chi, 2001). There were a total of 69 episodes. In addition to discussing the current problem state, some episodes focused on helping with the interface (such as labeling constructs), motivate and praise the student, suggest to try a more challenging problem, complete the session or help with technical problems (e.g. web browser suddenly closing). The maximum and the minimum number of episodes initiated by a tutor during a session was 20 and 4 respectively. Surprisingly, these 20 episodes occurred in a session of 1.5 hour duration which is not the longest session (the longest session lasted approximately 2 hours). In the session which consisted of 4 episodes, the first intervention occurred only in the 19th problem (the student completed 22 problems).

We are mainly interested in 37 episodes which discussed the current problem state or the relevant domain concepts. The following statistics were calculated using these 37 episodes. The average number of such episodes per tutor was 9.25. Five episodes contained a single utterance each, which was initiated by the tutor. For instance, a tutor utterance that occurred just after the completion of a problem was "Remember that the participation for weak entity is always total". The longest episode consisted of 9 utterances of which 4 were by the tutor. The student made more utterances than the tutor in only 2 episodes. Furthermore, only 2 episodes were student-initiated. This indicates that the tutor is more likely to be active in the interventions.

Only 20 (54%) episodes were considered to facilitate self-explanation. The criterion to label an episode as facilitating self-explanation is whether it discussed concepts that went beyond the current error, or facilitated justifications for the correct modelling decision. An example is presented in Figure 2, which occurred while the student was working on the problem shown in Figure 1. This dialogue contains two episodes, because it covers two different concepts (one related to weak entities, and the other one about total participation). Even though the tutor is leading the dialogue, the student has justified his decision for modeling GROUP as a weak entity

(*Student1*). Therefore, the dialogue provided an opportunity for the student to explicitly self-explain his modeling decision. Also, the student was able to identify and repair the misconception he had with weak entities and total participation after the tutor's explanation (*Tutor3*). Student's utterance about learning material during this session (*Student2*) can be considered as evidence of implicit self-explanation.

Tutor1:	Well done on changing GROUP to a regular entity!
Tutor2:	Use a weak entity only if its key is not unique in the database without its owner, i.e. in order for it to be unique, it has to be viewed with the owner's key as well. In this case, a group is unique without the student's id.
Student1:	I understand that, but I thought that group can't exist without students!?
Tutor3:	Exactly, and this is shown by the total participation in the relationship.
Student2:	I have learned something today

Fig.2. A dialogue from the study

The highest number of self-explanation dialogues in a session was 7, while the lowest was 2. As can be expected, the highest number of self-explanation dialogues occurred in the longest session. Even though all tutors initiated interaction episodes, two of them did not have any self-explanation episodes. This may be because the tutors were not explicitly asked to facilitate self-explanation, but to assist the students with problem solving.

When data was analysed to identify different strategies used by tutors, three strategies were prominent. Tutors were rephrasing feedback, providing problem-independent explanations and stating their observations before starting to discuss the problem state. The tutors who did not have any self-explanation eposides in their sessions mainly rephrased feedback to enable the student to understand their own mistakes. For example, the tutor prompted "Does AUTHOR need to be an entity?" or "The cardinalities of BORROWED_FROM needed fixing". Rephrasing feedback may have been effective because most students realised that the additional feedback was provided by a human observing their problem-solving process. If the self-explanation model is to repeat the same kind of prompting, it is difficult to ascertain whether it will have the same effect (Lepper, et al., 1993). The second strategy was to discuss the current problem state and then provide a problem-independent explanation. Figure 1 represents an example. These explanations provided an opportunity for the student to repair his/her mental model of the domain and generated further conversation. The third strategy used was to state the tutor's observations before starting to discuss the problem state. For example, tutor started the dialogue by saying "You seem to be having a few problems with relationships. Think about this. Can a student be enrolled in a course without involving a department?"

As the knowledge base in EER-Tutor is represented as a set of constraints, the errors were recorded as constraint violations. We analysed how frequently constraints were violated after related errors were discussed in self-explanation episodes, to see whether tutor interventions helped students to improve their knowledge. If these constraints represent psychologically appropriate units of knowledge, then learning should follow a smooth curve when plotted in terms of constraints (Anderson, 1993). To evaluate this expectation, the participants' logs were analysed, and each problem-state after a tutor invention in which a constraint was relevant was identified. These identified occasions are referred to as *occasions of application* (Mitrovic, et al, 2004). Each constraint relevance occasion was ranked 1 to *n*. For each occasion we recorded whether a relevant constraint was satisfied or violated. We then calculated the probability of violating a constraint on the first occasion of application, the second occasion and so on, for each participant. The probabilities were then averaged across all participants and plotted as a function of the number of occasions when a constraint was relevant (Figure 3).

As can be seen from Fig. 3.a, there is an outlier, increasing the probability of violating a constraint in the 4^{th} and the 5^{th} occasions. This is due to a single student violating the constraint dealing with total participation. For this student, the tutor provided a problem-independent explanation to help him identify and repair the misconception he had with weak entities and total participation (Figure 1). The explanation was not related to a problem state later on, as the discussion was a follow-up from another error related to weak entities. This may have been a reason for the subsequent violations of this constraint.

Figure 3.b shows the learning curve with the outlier removed. The probability of 0.22 for violating a constraint at its first occasion of application decreased to 0.02 at its eighth occasion of application, displaying a 90.9% decrease in the probability. The results of the mastery of constraints reveal that students seem to learn ER modelling concepts which were discussed by the tutors.

Twenty-eight different constraints were discussed in the self-explanation episodes. Three students did not violate any constraints in subsequent occasions after the tutor interventions. These students were tutored by three different tutors who followed strategies like rephrasing feedback, providing problem-independent explanations and stating tutor's observations at the beginning of the discussion. This suggests that all these strategies have been effective in helping the students learn domain concepts.

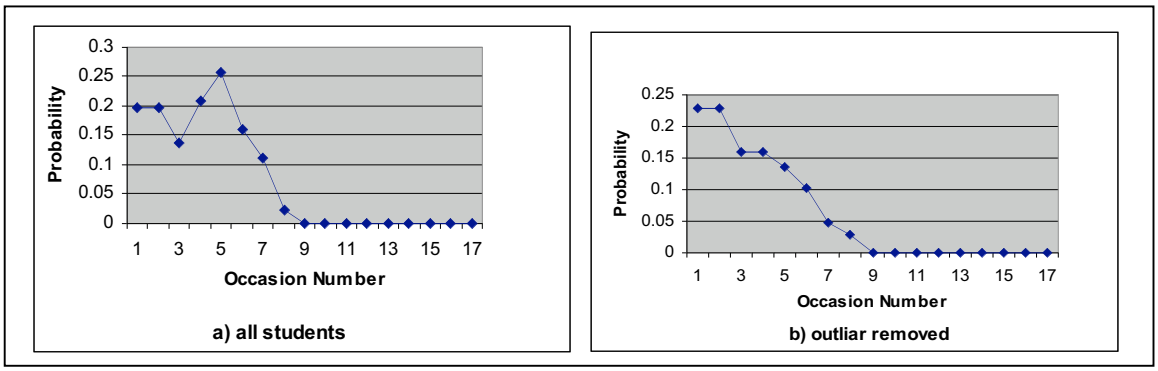


Fig. 3. Probability of violating a constraint as a function of number of occasions when that constraint was relevant

Timing of Interventions

The original version of EER-TUTOR provides feedback on demand, i.e. only when the student submits the solution. The tutors in this study also provided delayed feedback, which was well-received by the participants. Delayed feedback also provided an opportunity for students to correct the mistakes themselves. There were few instances where the student made a mistake and corrected it after referring the problem text again. For example, one of the problems required students to model CAR as an entity and *Colour* as a multi-valued attribute of CAR. The student modelled *Colour* as a simple attribute and then changed it to multi-valued as the last sentence in the problem text indicated that a car can have many colures. In such a situation, immediate feedback would not have been welcomed by the student, as he may have felt the intervention being intrusive.

The important issue with delayed feedback is how the tutors decided that the students needed help. In our study tutors provided help when the student (i) made the same type of mistake repeatedly (ii) asked for more help using the *More Help* button (iii) was inactive for some time, (iv) reacted to feedback, or (v) asked a problem-specific question through the chat interface. These scenarios will be discussed in detail in the next section.

Prototype of the Self-Explanation Model

The self-explanation model will be used to decide when to prompt to self-explain, what to self-explain and how to obtain self-explanations from learners. The model consists of three parts: error hierarchy, self-explanation dialogues and meta-constraints. Each component is now described in turn.

A hierarchy of errors was developed after analyzing different students' errors (Weerasinghe, 2003). A high-level view of the hierarchy is given in Figure 4. Nodes in this hierarchy are ordered from basic domain principles to more complicated ones. Violated constraints for each type of error are represented as leaves of the hierarchy. Some error types are further divided into sub errors. Constraints for some nodes are given in separate lines to indicate that each constraint corresponds to a specific error type. For instance, each constraint assigned to the node *Syntax Errors* specifies a different type of error. The error hierarchy enables the system to locate the appropriate dialogue to be used in case of incorrect student submission. If there are multiple errors, depth-first search identifies the dialogue to initiate with the student.

Self-explanation is facilitated through tutorial dialogues. We designed a tutorial dialogue for each type of error. Each dialogue consists of 4 stages. In the first stage, the model informs the student about the concept that s/he is having difficulty with, and then asks for a justification of the student's action. The purpose of the second stage is to assist the student in understanding why his action is incorrect. The third stage prompts the student to specify how to correct the mistake. A further opportunity is provided in the fourth stage to review the domain concept learned. An example of a tutorial dialogue is given in Figure 5. Initially, the system identifies the domain concept the student has problems with, and asks the student to explain it (*EERTutor1*). We have yet to decide how to obtain self-explanation from learners i.e. whether to provide a list of possible answers from which the correct one could be selected or to incorporate natural language processing module enabling learns to use free-form questions. If the student fails to provide the correct answer (*Student1*), s/he will be asked a more specific question that provides a further opportunity to understand the fundamental principle that is violated (*EERTutor2*). However, if s/he fails to correct the mistake even after going through a series of detailed questions, as the last resort the tutor will provide an explanation on how to the correct the mistake together with a brief description about the fundamental principle that needs to be learnt (*EERTutor5-7*). The dialogues use various types of interactions such as simple questions (*EERtutor1*), fill-in-a-blank (*EERTutor7*), or true-false questions, to motivate the student to self-explain. When a certain mistake is repeated, the model informs the student of its observations (*EERTutor1*), thereby providing an opportunity to reflect on his/her domain knowledge. As all

dialogues facilitate self-explanation by pointing out errors (*EERtutor3*), students are given opportunities to reflect on their problem solving procedure, which is another important meta-cognitive skill.

ALL ERRORS
- Syntax errors
  - 1 – More than one construct is used to model a word of the problem
  - 2 – Database does not have any entities
  - 3 – A connector is not connected to two entities
  - 4 – A connector is used to connect attributes is not a single line with no cardinality
  - 5 – An attribute is connected to more than one other attribute
  - 6 – An attribute is connected to an entity and another construct
  - 101 – An attribute is connected to a relationship and another construct
  - 7 – An entity is directly connected to another entity
  - 8 – An attribute is not connected to any other construct
  - 9 – A relationship is connected to another relationship
  - 10 – Names of entities and relationships are not unique
  - 21 – A relationship is not connected to two entities
- Semantic errors
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Connecting an attribute to an incorrect construct
  - Errors dealing with cardinalities and participation

Fig. 4. Overall view of the error hierarchy

Ideal SE behaviour will be represented as a set of meta-constraints, and will enable individualization of the self-explanation dialogues. As we discussed previously, the SE dialogues are pre-specified sequences of questions; however, for each individual student, the SE model will decide on the entry point to the dialogue, or the timing of the dialogue. The observations from the study will be used to develop meta-constraints.

As delayed feedback provided by the tutors in this study was well-received by the student participants, the self-explanation model will also provide delayed feedback. The critical issue then is to decide when it will be beneficial to intervene. As noted earlier, tutors intervened when the student (i) made the same type of mistake repeatedly, (ii) asked for more help using the *More Help* button, (iii) was inactive for some time, (iv) reacted to feedback, or (v) asked a problem-specific question through the chat interface.

When a student made the same error repeatedly, tutors provided a problem-independent explanation of the domain concept they have difficulty with. Some tutors initiated the dialogue by stating their observations. For instance, if it is difficult for the student to identify a weak entity, the tutor's initial response was "*You seem to be having some difficulty with regular entities. Let's look at regular entities in detail*." followed by an opportunity to discuss the corresponding domain concept. One meta-constraint will check for these situations, and will be violated when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake will be initiated, but the dialogue would start from the problem-independent question (*EERTutor1* in Figure 5).

As noted in (ii) (asking for more help using the *More Help* button), the student will be given an opportunity to receive more help for each feedback message provided by the system. If more help is requested, then the corresponding self-explanation dialogue will be initiated. For instance, if the student requested more help on CHAPTER being modeled as a regular entity and he has not made the mistake repeatedly, then dialogue will discuss the error within the current context (*EERTutor3* in Figure 5). Hence the self-explanation dialogues will be adaptive based on the student's domain knowledge and the self-explanation skills.

Even though all tutors intervened when a student has been inactive for about a minute, they tended to wait longer when the student is in the initial problem-solving phase (i.e. student has not submitted his solution and has not received any feedback from the system so far). One of the meta-constraint will identify that situation, by checking whether the student has made any attempts at the current problem, and has been inactive for a specified period of time. (such as 1.5 minutes, the time period we observed in the study). Violation of this constraint will initiate an evaluation of the student's solution even though it has not been submitted yet, and also prompt the student to identify which concept he is having difficulty with. For instance, if the student is having difficulty with the regular entity CHAPTER, and the evaluation of his solution identifies that it needs to be modeled as a weak entity, then the self-explanation model will help the student to understand the correct modeling decision through a series of questions. Figure 5 presents a sample dialogue that occurs between the student and the tutor in this case.

A student seems to be just reacting to feedback if he makes a single change without reflecting on the other changes that need to be carried out as a result. In ER modeling, certain changes trigger a number of other changes

in order to develop a syntactically correct model. For instance, if a regular entity with a key attribute is changed to a weak entity, then a partial key should be specified instead of the key attribute. Also one of the relationships of the regular entity needs to be changed to an identifying relationship etc. Sometimes students tend to make a single change suggested by the system and submit again for more feedback. This may lead to frustration and a feeling of exhaustion due to the number of attempts that the student has to go through to arrive at the correct solution. In such a situation, the student will be prompted to reflect on other related changes before submitting the solution. An example meta-constraint which identifies such situations is given below; when this constraint is violated, the system will remind the student to check whether other changes are necessary.

Relevance Condition: If the student has changed the solution to correct one mistake (i.e. violated constraint),
Satisfaction Condition: Then he should also make related changes.

For scenario (v) (asking problem-specific questions through the chat interface), the self-explanation model needs to enable learners to ask free-form questions while interacting with the system. We are currently investigating different alternatives. One alternative is to use TFIDF (Term Frequency Inverse Document Frequency) vector weighting scheme (Salton and Buckley, 1987) to retrieve information from a pre-defined question database. TFIDF is currently being used in another project by our group developing a question asking module while solving database mapping problems with the ERM-Tutor (Milik, et al., 2006). Another alternative is to incorporate a natural language processing module to the EER-Tutor.

EERTutor1: You seem to be having some difficulty with regular entities. Let's look at regular entities in detail. Can you tell me the general rule to decide whether something is a regular entity?
Student1*: I don't know*
EERTutor2: Ok! Let's try a simple thing. What is used to uniquely identify instances of a regular entity?
Student2*: The key attribute*
EERTutor3: Now tell me what is unique about *CHAPTER* regular entity?
Student3*: I'm not sure*
EERTutor4: Let me help you a bit. Check carefully whether any of the attributes of *Chapter* can uniquely identify the instances of *Chapter?*
{After sometime} Please tell me whether you managed to find any attributes
Student4: *No*
EERTutor5: Fine. It doesn't have any attributes that can uniquely identify instances of *Chapter*. So now can you tell me why modeling *Chapter* as a regular entity is incorrect?
Student5 : *Because it doesn't have a key attribute.*
EERTutor6: Great! Now tell me the best way to model *CHAPTER*?
Student6*: A weak entity*
EERTutor7: Well Done! Now select the correct pair of words to fill in the blanks so that you'll remember what you learnt.
A ….. is used to uniquely identify instances of a …….. entity.
Student7*: key attribute ,regular*
EERTutor8: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 5. A sample tutorial dialogue

The short-term student model in EER-Tutor consists of a list of satisfied and a list of violated constraints for the student's solution, while the long-term model records the history of each constraint (Zakharov et al., 2005). In the enhanced version of EER-Tutor, the long-term model will additionally record details of the self-explanation skill of a student in terms of types of errors made (in terms of meta-constraints) and the level of prompting the student needed to correct his mistake for every constraint.

Conclusions and Future Work

Self-explanation is an effective learning strategy to facilitate deep learning. This research focuses on developing a self-explanation model for both ill- and well-defined tasks. As the first step, we conducted a preliminary study to observe how tutors prompt students to guide them towards solutions while using EER-Tutor, a constraint-based tutoring system for learning Entity-Relationship modelling. In addition to the feedback received by the system, the students were prompted by the tutors through a chat interface. Students also had the opportunity to initiate a dialogue with the tutor either through the chat interface or using the *More Help* button.

The interactions between the tutors and the students were analysed to identify the different episodes, each pertaining to a single topic. Only 20 (54%) of these episodes that discussed the current problem state or the relevant domain concepts were considered to facilitate self-explanation. These episodes either facilitated the discussion of domain concepts that went beyond the current error, or prompted justifications for the correct

modelling decision. The user logs were analysed to investigate how frequently a certain error occurred after each self-explanation episode. The analysis indicated that in spite of different tutoring strategies, the tutor interventions helped the learners to improve their understanding of ER modelling concepts.

The findings from the reported study are being used to develop the self-explanation model for the EER-Tutor. The next step is to incorporate the model into the EER-Tutor, and evaluate it in an authentic classroom environment. We will then implement the same SE model in an ITS for a well-defined task.

References

Aleven, V., Koedinger, K. R., Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Artificial Intelligence in Education, Lajoie, S.P. and Vivet, M.(eds.), IOS Press (1999) 199-206.

Anderson, J. R., Rules of the mind. Erlbaum, Hillsdale, NJ, 1993.

Baghaei, N., Mitrovic, A., Irwin, W. A Constraint-Based Tutor for Learning Object-Oriented Analysis and Design using UML. In: C.K. Looi, D. Jonassen, M. Ikeda (eds), ICCE 2005, 11-18.

Chen, P. (1976). The ER Model - Toward a Unified View of Data. ACM Transactions Database Systems, 1(1), 9-36.

Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. Advances in Instructional Psychology, (2000) 161-238.

Chi, M.T.H., Bassok, M., Lewis, W., Reimann, P., Glaser, R., Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. Cognitive Science, 13 (1989), 145-182.

Chi, M. T. H., S. A. Siler, et al. Learning from human tutoring. Cognitive Science 25 (2001), 471-533.

Conati, C., VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. Int. J. Artificial Intelligence in Education, 11 (2000) 389-415.

Goel, V., Pirolli, P. Motivating the Notion of Generic Design with Information Processing Theory: the Design Problem Space. AI Magazine, 10 (1988) 19-36.

Goel, V., Pirolli, P. (1992) The Structure of Design Problem Spaces. Cognitive Science, 16, 395-429.

Graesser, A. C., Person, N. et al. Colloborative dialog patterns in naturalistic one-on-one tutoring. Applied Cognitive Psychology, **9**, (1995) 359-3

Lepper, M.R., Woolverton, M. Mumme, D. L., and Gurtner, J.L. Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In Lajoie, S.P. and Derry, S. J. (eds.) Computer as Cognitive Tools, Lawrence Erlbaum, Hillsdale, New Jersey, (1993)75 -105.

Milik, N., Marshall, M., Mitrovic, A. Teaching Logical Database Design in ERM-Tutor. In: M. Ikeda & K. Ashley (eds) Proc. ITS 2006.

Mitrovic, A., Ohlsson, S., Evaluation of a Constraint-Based Tutor for a Database Language. Int. J. Artificial Intelligence in Education, 10 (1999), 238-256.

Merrill, D. C., Reiser, B. et al. Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. Journal of the Learning Sciences 2(3) (1992) 277-305.

Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. Journal of Interactive Learning Research, 15(4), (2004) 409-432.

Reitman, W.R. (1964) Heuristic Decision Procedures, Open Constraints, and the Structure of Ill-defined Problems. In: M.W. Shelly, G.L. Bryan (eds) Human Judgements and Optimality. New York, Wiley.

Salton, G., Buckley, C., Term Weighting Approaches in Automatic Text Retrieval. Technical Report #87-881 Computer Science Dept, Cornell University, Ithaca, NY, 1987.

Suraweera, P. and Mitrovic, A., An Intelligent Tutoring System for Entity Relationship Modelling. *Int. J. Artificial Intelligence in Education*, v14n3-4, 375-417, 2004.

Weerasinghe, A. Exploring the effects of Self-Explanation in the Context of a Database Design Tutor. MSc Thesis, University of Canterbury, 2003.

Weerasinghe, A., Mitrovic, A. Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. Int. J. of Knowledge-based and Intelligent Engineering Systems, 10(1),(2006) 3-19.

Zakharov, K., Mitrovic, A., Ohlsson, S., Feedback Micro-engineering in EER-Tutor. In: Looi, C-K., McCalla, G., Bredeweg, B., Breuker, J. (eds.) Proc. Artificial Intelligence in Education AIED 2005, IOS Press, (2005) 718-725.

De Vicente, A., Pain, H. Informing the detection of the students' motivational state: An empirical study. In: Cerri, S.A., Gouarderes, G., Paraguacu, F. (eds.), Proc. 6th Int. Conf. Intelligent Tutoring Systems (2002) 933-943.

Towards a General Model for Supporting Explanations to Enhance Learning

Amali WEERASINGHE, Antonija MITROVIC and Brent MARTIN
Intelligent Computer Tutoring Group
University of Canterbury, Christchurch, New Zealand

Abstract. We present a project with the goal of developing a general model for supporting explanations, which could be used in both well- and ill-defined instructional tasks. We have previously studied how human tutors provided additional support to students learning with an existing intelligent tutoring system. Analysis of the interactions by human tutors indicates that they have helped the students to improve their understanding of database design. This paper presents the explanation model developed based on these findings.

Introduction

Even though intelligent tutoring systems (ITS) have been very effective in supporting students' learning, some students attempt to *game the system* in order to complete the problems [2]. As a result, those students acquire shallow knowledge which is sufficient only to pass exams not to solve transfer problems successfully. Self-explanation (SE) has been shown to facilitate deep learning [3]. Several ITSs were enhanced with self-explanation support in domains such as physics [4], mathematics [1], database design [8] and data normalization [6]. All these instructional tasks except database design are well-defined, as problem solving is well structured, and therefore self-explanation expected can be clearly defined. Database design is an ill-defined task: the final result is defined in abstract terms, but there is no algorithm to find it.

Our long-term goal is to develop a model of explanation which will provide adaptive support across domains. The main objective of this model is to assist students to develop their self-explanation skills while being prompted to explain their mistakes to the tutor. Since we previously implemented explanation support for the database design tutor [8], the initial work started with the same tutor. As the first step, we conducted an observational study [7] focusing on how students interacted with EER-Tutor [6], while getting additional help from a human tutor through a chat interface. A model to facilitate explanations was then developed. This model is presented in the next section followed by future work.

1. Prototype of the Self-Explanation Model

The explanation model will be used to decide when to prompt for explanations, what to explain and how to obtain explanations from learners. The model consists of three parts: error hierarchy, tutorial dialogues and rules for adapting them. Each component

is now described in turn. Error hierarchy and the dialogues are used to determine timing and content of the explanations respectively. Learners will be able to provide explanations by selecting the correct one from a list provided by the tutor.

Error Hierarchy: The domain model of constraint-based tutors is represented as a set of constraints [6]. Violations of constraints indicate mistakes in students' solutions. In previous work, we developed a hierarchy of errors students make in the Entity-Relationship (ER) domain [8], which categorizes errors as being syntactic or semantic in nature. Syntax errors are simple, each requiring only one feedback message to be given to the student; for that reason, every syntactic error corresponds to a particular constraint being violated. There were 12 such constraints. For example, constraint 8 is violated when the student creates an attribute which does not belong to an entity/relationship type. The hierarchy for semantic errors is deeper, with error types further divided into sub-errors: (i) *Using an incorrect construct type*, (ii) *Extra constructs*, (iii) *Missing constructs*, (iv) *Connecting an attribute to an incorrect construct* and (v) *Errors dealing with cardinalities and participation*. These nodes are ordered from basic domain principles to more complicated ones. Violated constraints for each type of error are represented as leaves of the hierarchy.

The ER error hierarchy was developed for a particular domain, so we were interested whether it can be reused in other domains. With that goal, we tried to fit the errors for ER-to-relational-mapping, data normalization and fraction addition. ER-to-relational mapping involves mapping an ER schema to a relational schema using the 7-step mapping algorithm [5]. Data normalization is the process of refining a relational database schema in order to ensure that all relations are of high quality. All these tasks are well-defined, due to the deterministic algorithms used.

During this investigation, several refinements were identified. The highest level of the refined error hierarchy has two nodes: *Basic syntax errors* and *Errors dealing with the main problem solving activity*. The second node is now further categorized into five nodes: (i) *Using an incorrect construct type* (ii) *Extra constructs* (iii) *Missing constructs* (iv) *Associations* and (v) *Failure to complete related changes*. The subsequent levels deal with domain-specific concepts. The common feature in all these tasks is that the syntactic and semantic accuracy of a solution can be completely evaluated by the components of the solution and its associations. However, there are exceptions. For instance, in reading and comprehension, where learners are asked to answer questions based on a paragraph, the accuracy of an answer cannot be evaluated by checking only for the correct words according to the grammatical rules. We also need to understand the implicit semantic meaning of the sentence. Therefore, our error hierarchy is not useful in such cases. In summary, we have been able to use this hierarchy in four different types of tasks: thus we believe it would be sufficiently general to be used for different types of instructional tasks only when the solution can be completely evaluated by the components of the solution and its associations.

Tutorial Dialogues: In our model, explanation is facilitated through tutorial dialogues. We present a summary here, for more details see [7]. For each error type (i.e. each leaf node in the hierarchy), we designed a dialogue consisting of four stages. In the first stage, the dialogue informs the student about the concept that s/he is having difficulty with. For example, *You seem to be having some difficulty with regular entities. Let's look at regular entities in detail. Can you tell me the general rule to decide whether something is a regular entity?* (Prompt1). The purpose of the second stage is to assist the student in understanding why the performed action is incorrect. An instance of such a prompt is *Now tell me what is unique about CHAPTER regular*

entity? (Prompt 2). The third stage prompts learners to specify how to correct the mistake. In the fourth stage, the student can review the domain concept learned. Although the prompts are domain-specific, the structure of the dialogues is domain-independent. We are currently investigating the applicability of the dialogue structure to various domains.

Rules for adapting dialogues: These rules enable individualization of the dialogues. For each student, the rules decide on the entry point into the dialogue, and/or the timing of the dialogue. Currently there are eight rules and they are based on the study [7]. For example, rule 4, dealing with customizing the entry point to the dialogue, is executed when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake is initiated, but the dialogue starts from the problem-independent question (Prompt 1 above). If the error was made less than *n* attempts, then the dialogue starts from the error within the current context (Prompt 2).

Rule 1 (dealing with timing of dialogues) checks whether the student made any attempts at the current problem, and has been inactive for a specified period of time (such as 1.5 minutes, the time period we observed in the study [7]). This rule will initiate an evaluation of the student's solution even though it has not been submitted yet, and start a dialogue to discuss the most suitable error (depending on the error hierarchy and the student solution). Individualization of the chosen dialogue will be based on the rule 4 discussed above. As these rules do not depend on domain-specific details to individualise dialogues, they can be used across domains.

2. Conclusions and Future Work

Self-explanation is an effective strategy to facilitate deep learning. This research focuses on developing a model for supporting explanations for both ill-defined and well-defined tasks. This model is based on the findings of an initial study using EER-Tutor. The next step is to incorporate the model into both EER-Tutor and NORMIT. These enhanced systems will later be evaluated in authentic classroom environments.

References

1. Aleven, V., Koedinger, K. R., Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Lajoie, S., Vivet, M.(eds.), Proc. AIED 1999, IOS Press, 1999, pp. 199-206.
2. Baker, R.S., Roll, I. Corbett, A.T., Koedinger, K. R. Do Performance Goals Lead Students to Game the System In: Looi, C-K., McCalla, G., Bredeweg, B., Breuker, J. (eds.) Proc. AIED 2005, IOS Press, 2005, pp. 57-64.
3. Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. Advances in Instructional Psychology, 2000, 161-238.
4. Conati, C., VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. Artificial Intelligence in Education, 11, 2000, 389-415.
5. Milik, N., Marshall, M., Mitrovic, A. Teaching Logical Database Design in ERM-Tutor. In: M. Ikeda & K. Ashley (eds). Proc. 8th Int. Conf. on Intelligent Tutoring Systems, 2006, pp. 707-709.
6. Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. Interactive Learning Research, 15(4), 2004, 409-432.
7. Weerasinghe, A., Mitrovic, A. Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based Tutors, Aleven, V., Ashley, K., Lynch, C. and Pinkwart, N. (eds.), Workshop on ITS for Ill-defined domains at ITS2006, 2006, pp. 56-64.
8. Weerasinghe, A., Mitrovic, A. Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. Knowledge-based and Intelligent Engineering Systems, 10(1), 2006), 3-19.

Developing a General Model for Supporting Self-Explanation for Intelligent Tutoring Systems

Amali Weerasinghe
Intelligent Computer Tutoring Group
University of Canterbury, Christchurch, New Zealand
acw51@cosc.canterbury.ac.nz

Abstract. We present a project with the goal of developing a general model for self-explanation (SE) support, which could be used in both well- and ill-defined instructional tasks. A model to support SE was developed based on the findings of a preliminary study using an existing intelligent tutoring system that teaches database design. We used this model in a Wizard-of-Oz study, to provide adaptive SE support. The results show that students did learn the relevant domain knowledge. Human tutors mostly agreed with the interventions generated by the model.

1. Introduction

Self-explanation (SE) has been shown to be highly effective in assisting students to acquire deep knowledge that help them apply their knowledge to novel situations. [2]. During the SE process students attempt to explain the new information to themselves as a way of understanding it better. Several intelligent tutoring systems (ITSs) were enhanced with SE support in domains such as physics [4], mathematics [1], and database design [8]. With the exception of database design, all these domains are well-defined, as problem solving is well structured, and therefore SE expected can be clearly defined. Database design is an ill-defined task: the final result is defined in abstract terms, but there is no algorithm to find it. Although the above ITSs improved student performance, none of these models have been used for both ill- and well-defined tasks.

Our long-term goal is to develop a general model to provide adaptive SE support across domains. Since we previously implemented SE support for the database design tutor [8], the initial work started with the same tutor. As the first step, we conducted an observational study [7] focusing on how students interacted with EER-Tutor [6], while getting additional help from a human tutor through a chat interface. A model to support SE was developed based on this study. This model addresses there basic decisions: when to prompt for self-explanation, what to self-explain and how to obtain explanations from learners. In order to understand the applicability of the model in a well-defined domain, another observational study was conducted, this time with the ERM-Tutor, an ITS for ER-to-relational mapping [5].

We present the model in Section 2. The next section describes the observational study and results. Conclusions and future plans are presented in the last section.

2. Prototype of the Self-Explanation Model

The model determines the timing and content of self-explanations. The model consists of three parts: error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. SE is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and SE skills. Learners are able to provide explanations by selecting the correct option from a list provided by the tutor. Each component is now described.

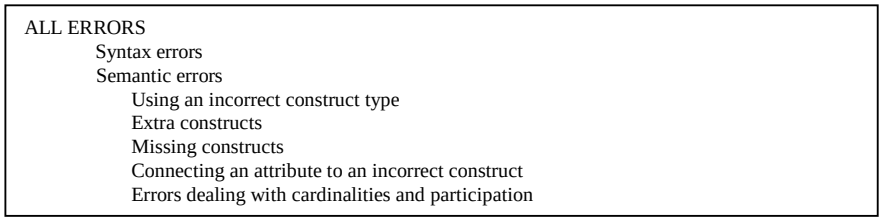


Fig. 1. Overall view of the error hierarchy

2.1 Error Hierarchy

In previous work, we developed a hierarchy of errors in the Entity-Relationship (ER) domain [8], which categorizes errors as being syntactic or semantic in nature. A high-level view of the hierarchy is given in Figure 1, with nodes ordered from basic domain principles to more complicated ones. Violated constraints for each type of error are represented as leaves of the hierarchy. Syntax errors are simple, each requiring only one feedback message to be given to the student; for that reason, every syntactic error corresponds to a particular constraint being violated. For example, constraint 8 is violated when the student creates an attribute which does not belong to an entity/relationship type. The hierarchy for semantic errors is deeper, with error types further divided into sub-errors. The ER error hierarchy was developed for that particular domain, so we were interested whether it can be reused in other domains. With that goal, we tried to fit the errors in the domains of ER-to-relational mapping, data normalization and fraction addition to this structure. ER-to-relational mapping involves mapping an ER schema to a relational schema using the 7-step mapping algorithm [5]. Data normalization is the process of refining a relational database schema in order to ensure that all relations are of high quality. Even though fraction addition is a simple domain it is quite different from the other domains that we have explored. All these tasks are well-defined because of the deterministic algorithms used.

During this investigation, several refinements were identified. There were situations when it was not enough to present a single feedback message for some violated syntax constraints: a dialogue was required. Therefore, we modified the structure of the hierarchy to divide all error types into two main categories: *Basic Syntax Errors* and *Errors dealing with the main problem-solving activity*. Under the

new node *Basic Syntax errors*, we included simple syntax errors, such as checking whether the student has filled the required fields, the components used to fill the required fields are valid etc. Hence it is sufficient to discuss such errors using a single message. The other category requires a dialogue to be conducted. The second node is now further categorized into five nodes: (i) *Using an incorrect construct type* (ii) *Extra constructs* (iii) *Missing constructs* (iv) *Associations* and (v) *Failure to complete related changes*. The subsequent levels deal with domain-specific concepts. The common feature in all these domains that we explored is that the syntactic and semantic accuracy of a solution can be completely evaluated by the components of the solution and its associations. However, there are exceptions. For instance, in reading and comprehension, where learners are asked to answer questions based on a set of paragraphs, the accuracy of an answer cannot be evaluated by checking only for the correct words according to the grammatical rules. We also need to understand the implicit semantic meaning of the sentence. Therefore, our error hierarchy is not useful in such cases. In summary, we have been able to use this hierarchy in four different types of tasks: thus we believe it would be sufficiently general to be used for different types of instructional tasks only when the solution can be completely evaluated by the components of the solution and its associations.

2.2 Tutorial Dialogues

In our model, explanation is facilitated through tutorial dialogues. We present a summary here, for more details see [7]. A hypothetical dialogue is given in Figure 2.

EERTutor1: You seem to be having some difficulty with regular entities. Let's look at regular entities in detail. Can you tell me the general rule to decide whether something is a regular entity?
Student1: *I don't know*
EERTutor2: Let's try a simple thing. How do we model characteristics of an entity?
Student2: *Using Attributes*
EERTutor3 : Great ! Now tell me why modelling *Grade* as an entity is incorrect?
Student3: *I'm not sure*
EERTutor4: Let's see …Check carefully whether you can find any attributes for *Grade*.
{After sometime} Pls. tell me whether you manage to find any attributes for *Grade*
Student4: *No*
EERTutor5: Fine. It doesn't have any attributes. So now can you tell me why modelling *Grade* as an entity is incorrect?
Student5: *Because it doesn't have any attributes to describe its characteristics*
EERTutor6: Great! Now tell me the best way to model *Grade*?
Student6: *I need more help*
EERTutor7: Ok. If *Grade* doesn't have attributes, it cannot be an entity. So it should either be a relationship or an attribute. Now tell me the best way to model *Grade?*
Student7: *Attribute*
EERTutor8: Great ! Now use the correct word to fill in the blanks so that you'll remember what you learnt. …….. are used to describe characteristics of an …….. in a ER model
Student8: *attributes, entity*
EERTutor9: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig.2. A hypothetical dialogue

For each error type (i.e. each leaf node in the hierarchy), we designed a dialogue consisting of four stages. In the first stage, the dialogue informs the student about the

concept that s/he is having difficulty with (*EERTutor1*). The purpose of the second stage is to assist the student in understanding why the performed action is incorrect (*EERTutor3*).The third stage prompts learners to specify how to correct the mistake (*EERTutor6*). In the fourth stage, the student can review the domain concept learned (*EERTutor8*). Although the prompts are domain-specific, the structure of the dialogues is domain-independent. We are currently investigating the applicability of the dialogue structure to various domains.

2.3 Rules for Adapting Dialogues

Adaptation rules enable individualization of the tutorial dialogues, by using the student model to decide on the timing, selection and entry point into the dialogue. Currently there are six rules and they are based on the findings of the previous study [7]. For example, rule 4, dealing with customising the entry point of the dialogue, is activated when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake is initiated, but the dialogue starts from the problem-independent question (such as *EERTutor1* in Figure 2). If the error was made less than *n* attempts, then the dialogue starts from the error within the current context (*EERTutor3* in Figure 2). As these rules do not depend on domain specific details to individualise dialogues, the rules can be used across domains.

3. Observational study

We conducted an experiment with the ERM-Tutor in April 2006 at the University of Canterbury, which involved volunteers from a database course and experienced tutors. Two types of feedback were provided: typical feedback provided by the system, and SE feedback provided by the model. The study was conducted as a Wizard-of-Oz study, in which the first author simulated the actions of the model. This additional assistance was given through a chat interface, and will be referred to as interventions hereinafter. The first author used the dialogues from a written script. However, it was not always possible to use the scripted prompts in the later stages of dialogues because the student answers were not constrained as in the proposed system.

Participants interacted with ERM-Tutor in one room, while the first author observed from another room. The participants could initiate interventions through the chat interface or the *More Help* button. Participants were expected to use the system for at least an hour. However, students themselves decided when to end the session. At the end of the session, participants filled out a questionnaire. The first phase of the study involved analysing the logs to investigate the effectiveness of the SE episodes. In the second phase, human tutors were asked to judge the appropriateness of interventions by observing recorded sessions. A time line indicating all the interventions was provided to the judges, who indicated whether they agreed with the timing and the content of interventions. When they disagreed, they were requested to provide justifications.

Ten students and five professional tutors (acting as judges) participated in the study. The average session duration was 59 minutes (sd=15.3). The average number of problems attempted and completed was 11 (sd = 4.6), and 8.4 (sd =5.2) respectively.

From the logs, we identified 65 episodes, each pertaining to a single topic [3]. In addition to facilitating SE, some episodes focused on helping with the interface, completing the session or helping with technical problems. The number of episodes per session ranged from 1 to 13, with a mean of 6.5 (sd = 4.3). We are mainly interested in 31 episodes which facilitated SE. An example SE episode is given in Figure 3. In this dialogue, the student is incorrectly applying step 4 of the mapping algorithm to the identifying relationship, while that step should only be applied to regular relationship types. The correct action here is simply to move to the next step. In this situation, the model aims to assist the student to understand that this step is not necessary.

ermtutor: what do you need to do when you're mapping a 1:N relationship?
coscstudent001: map the n-cardinality table
ermtutor: yes, what is the attribute that needs to be included
coscstudent001: the code from course
ermtutor: yes, good. but can you see this is a special case?
coscstudent001: because section is a weak entity?
ermtutor: yes

Fig 3. A dialogue from the study

In order to investigate whether the SE episodes were effective, we analysed how frequently the same error occurred after each episode. As the knowledge base in ERM-Tutor is represented as constraints, the errors were recorded as violations of constraints. Thus we analysed how frequently the constraints that were discussed in the SE episodes were violated subsequently. However, some students were able to correct the mistake themselves just before the episode started. In another situation, a student indicated that he did not require any assistance (even after a period of inactivity) when prompted. Also, some violated constraints were not consistent with the actual mistake in the student solution due to a coding problem. It was not possible to specify a constraint for such episodes. Those episodes were excluded from the analysis. The remaining 15 (48.3%) episodes were included in this analysis. The learning curve for these constraints (Figure 4) is not smooth due to the small sample size. i.e. this analysis involved only 7 constraints discussed in 15 SE episodes with 7 participants. The students seem to learn domain concepts which were discussed in the SE episodes.

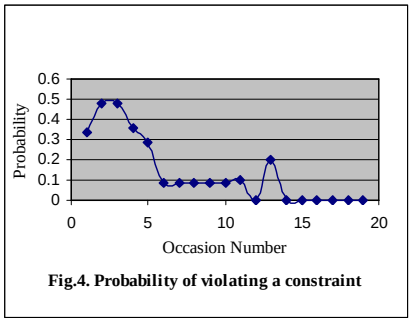


Fig.4. Probability of violating a constraint

In phase 2, five judges analyzed the interventions and indicated whether they agreed with their timing and content. Number of sessions analyzed ranged from 1 to 3 per judge. All SE episodes were categorised by the rules that initiated them. Five rules were relevant. Rule 4 (Section 2.3) was relevant in 21 episodes and had the highest number of disagreements. Judges disagreed in 7 (33.3%) occasions. Timing was the issue in six instances and judges wanted to intervene earlier. They disagreed with the content in three situations. For instance, a judge suggested "Is there a regular 1:N relationship to map in this problem?" instead of the first prompt in Figure 3.

One of the issues to be addressed is how to effectively facilitate SE when nothing needs to be done in a particular step (Figure 3). According to our model, the initial prompt was "*What do you need to do when you're mapping a 1:N relationship?*" which may imply that the student needs to perform an action, even if there is nothing to do. It would be better if the prompt is changed to "*Do you know which type of relationship needs to be mapped in this step*?".

Some judges preferred earlier interventions than those suggested by the model. The model waits for 3 repeated mistakes before initiating a SE episode. However, it might be effective to intervene after two repeated mistakes, because it is easier to assess what the student is trying to achieve in this particular domain. As the result, the number of times a mistake needs to be repeated may be domain-dependent. Also the time period of inactivity that the model waits before intervening may also need to be domain-dependant; however, there was no disagreement on these. Further investigation is needed before the model is changed.

4. Conclusions and Future Work

Self-explanation is an effective strategy to facilitate deep learning. This research focuses on developing a model to support SE for both ill- and well-defined domains. A prototype model was developed based on the findings of a preliminary study using EER-Tutor, a database design tutor. This paper focuses on the study which used the prototype model with ERM-Tutor, an ITS teaches ER-to-relational mapping. In addition to the feedback provided by the system, SE was facilitated through a chat interface. The interventions through the chat interface were based on the model.

Analysis of user logs indicates that students learn domain concepts discussed in the SE episodes. Human tutors who were asked to analyse the episodes mostly agreed with the interventions generated by the model. The findings from this study are currently being used to refine the model. The next step is to incorporate the model into both EER-Tutor and ERM-Tutor and evaluate them in authentic classroom environments.

References

1. Aleven, V., Koedinger, K. R., Cross, K. Tutoring Answer Explanation Fosters Learning with Understanding. In: Lajoie, S., Vivet, M.(eds.), Proc. AIED 1999, IOS Press, 1999, pp. 199-206.
2. Chi, M. T. H. Self-explaining Expository Texts: The dual processes of generating inferences and repairing mental models. *Advances in Instructional Psychology*, 2000, 161-238.
3. Chi, M. T. H., S. A. Siler, et al. Learning from human tutoring. *Cognitive Science* 25, 2001, 471-533.
4. Conati, C., VanLehn, K. Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation. *Artificial Intelligence in Education*, 11, 2000, 389-415.
5. Milik, N., Marshall, M., Mitrovic, A. Teaching Logical Database Design in ERM-Tutor. In: M. Ikeda & K. Ashley (eds). Proc. 8th Int. Conf. on Intelligent Tutoring Systems, 2006, pp. 707-709.
6. Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. *Interactive Learning Research*, 15(4), 2004, 409-432.
7. Weerasinghe, A., Mitrovic, A. Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based Tutors, Aleven, V., Ashley, K., Lynch, C. and Pinkwart, N. (eds.), Workshop on ITS for Ill-defined domains at ITS2006, 2006, pp. 56-64.
8. Weerasinghe, A., Mitrovic, A. Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. *Knowledge-based and Intelligent Engineering Systems*, 10(1), 2006), 3-19.

A Preliminary Study of a General Model for Supporting Tutorial Dialogues

Amali WEERASINGHE, Antonija MITROVIC and Brent MARTIN
Intelligent Computer Tutoring Group
University of Canterbury, Christchurch, New Zealand
{acw51, tanja, brent}@cosc.canterbury.ac.nz

Abstract: One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. We present a project with the goal of developing a general model for supporting tutorial dialogues that could be used in both well- and ill-defined instructional tasks. We have previously studied how human tutors provide additional support to students learning with an existing intelligent tutoring system. On the basis of these findings we developed a model for supporting tutorial dialogues, which we present in this paper. We used this model in a Wizard-of-Oz study to provide adaptive support. The results show that students did learn the relevant domain knowledge and that human tutors mostly agreed with the interventions generated from the model.

Keywords: tutorial dialogues, constraint-based tutors

1. Introduction

The main objective of the Intelligent Tutoring Systems (ITS) research community is to develop tutoring systems to achieve the effectiveness of one-on-one human tutoring, the most effective form of instruction [2]. One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. These dialogues provide opportunities for students to reflect on the existing knowledge and to construct new knowledge. Some of the dialogue-based tutoring systems that have been developed are Why2-Atlas [5], Auto Tutor [5], CIRCSIM-Tutor [7] and Geometry Explanation Tutor [1]. Why2-Atlas and AutoTutor use the dialogues as the main activity to help students learn the domain knowledge. The other systems provide problem-solving environments as the main activity and use tutorial dialogues as a way of remediating errors in the student solutions. For example, CIRCSIM-Tutor is a natural language (NL) tutor that helps students solve a set of problems in cardiovascular physiology relating to regulation of blood pressure. The Geometry Explanation Tutor requires students to justify the problem-solving steps in their own words. All these instructional tasks are well-defined: problem solving is well structured, and therefore explanations expected from learners can be clearly defined. In contrast, database design is an ill-defined task: the final result is defined only in abstract terms, and there is no algorithm to find it. In previous work we incorporated tutorial dialogues to our database design tutor [10].

Our long-term goal is to develop a general model for supporting dialogues across domains. Since we previously implemented dialogues for EER-Tutor [8], the initial work on this project started with the same system. As the first step, we conducted an observational study [9] focusing on how students interacted with EER-Tutor [8], while getting additional help from a human tutor through a chat interface. From the results of this study, we

developed a model to support dialogues, which we present in Section 2. In order to investigate the applicability of the model in a well-defined domain, another observational study was conducted, this time with the ERM-Tutor [6]. Section 3 describes the observational study and the results. Conclusions are given in the final section.

2. Prototype of the Model

Tutorial dialogues have been used in different pedagogical contexts based on the domain and the target student group. However, the problem-solving tasks supported were well-structured, and the types of explanations expected from students can be clearly defined. For example, in Mathematics and Physics, students are expected to explain the theorems that they have used. However, it is challenging to incorporate dialogues in an open-ended domain such as database design. It is not sufficient to ask the students to explain the concepts of database modeling, as the database design skills can only be developed through extensive practice. We also believe prompting them to explain every solution step will potentially place a heavy cognitive burden on the students. This may also demotivate natural explainers from using the dialogues. Hence tutorial dialogues are used to remediate errors in the student solution.

Our model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level an error type is associated with one or more violated constraints, which form leaves of the hierarchy. The error types are then grouped into higher-level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue learners are able to provide answers by selecting the correct option from a list provided by the tutor. Each component is now described in detail.

2.1 Error Hierarchy

In previous work we developed a hierarchy of errors students make in the Entity-Relationship (ER) domain [10], which classifies all errors into syntactic and semantic in nature. A high level view of the hierarchy is given in Figure 1, showing the top three levels only. Syntax errors are generally simple, each requiring only one feedback message to be given to the student rather than initiating a dialogue; for that reason, every syntax error corresponds to a single violated constraint. For example, 58 constraints are associated with syntax errors for the ER domain; constraint 7 is violated when two entities are directly connected to each other. In contrast, the hierarchy for semantic errors is deeper because constraints are often related by some high-level concept. For semantic errors, error types are further divided into sub-errors (Figure 1) .

We were interested to investigate whether this hierarchy can be reused in other domains. For this investigation, we used the constraints from three constraint-based tutors: ERM-Tutor, which teaches mapping conceptual database schemas into relational ones [6], NORMIT which teaches data normalization and a fraction addition tutor. ERM-Tutor teaches logical database design which involves mapping an ER-schema to a relational schema using the 7-step mapping algorithm [4]. Data normalization is the process of refining a relational database schema in order to ensure that all relations are of high quality

[8]. All three domains are well-defined, because of the existence of algorithms to carry out each task.

ALL ERRORS
- Syntax errors
- Semantic errors
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Connecting an attribute to an incorrect construct
  - Errors dealing with cardinalities and participation

Fig. 1. Overall view of the error hierarchy

During this investigation, we identified situations when it was not enough to present a single feedback message for some violated syntax constraints: a dialogue was required. Therefore, we modified the structure of the error hierarchy to divide all error types into two main categories: *Basic Syntax Errors* and *Errors dealing with the main problem-solving activity* (Figure 2). Under the new node *Basic Syntax errors*, we included simple syntax errors, such as checking whether the student has filled the required fields, the components used to fill the required fields are valid etc. Hence it is sufficient to discuss such errors using a single message. The other category requires a dialogue to be conducted.

Another refinement required was to make the two domain-specific nodes *Connecting an attribute to an incorrect construct* and *Errors dealing with cardinalities and participation* more general so that the overall hierarchy can be used across domains. As these two nodes deal with associations between solution components, it is appropriate to have a new node *Associations* (Figure 2). This new node has different domain-specific children. For the ER domain, *Connecting an attribute to an incorrect construct* and *Errors dealing with cardinalities and participation* are child nodes of *Associations.*

The final refinement was made based on an observation from the previous study [9]: some students seem to be reacting to feedback on errors by making suggested changes without reflecting on other modifications that also need to be carried out. In ER modeling if a regular entity with a key attribute is changed to a weak entity a partial key should be specified instead of the key attribute. This may lead to frustration due to the number of attempts that the student has to go through to arrive at the correct solution. A new node *Failure to complete related changes* was added to the existing error hierarchy, which reminds the student to check whether other changes are necessary (Figure2). In such cases, the student will be prompted to reflect on other related changes before submitting the solution.

ALL ERRORS
- Basic syntax errors
- Errors dealing with the main problem solving activity
  - Using an incorrect construct type
  - Extra constructs
  - Missing constructs
  - Associations
  - Failure to complete related changes

Fig. 2. Overall view of the refined error hierarchy

Figure 2 only shows the top three levels of the error hierarchy; these levels are domain-independent and the lower levels deal with domain-specific concepts. The common feature in all these tasks is that the syntactic and semantic correctness of a solution can be completely evaluated by examining the components of the solution and their associations.

However, there are exceptions. For instance, in reading and comprehension, where learners are asked to answer questions based on a paragraph, the accuracy of an answer cannot be evaluated by checking only for the correct words according to the grammatical rules. We also need to understand the implicit semantic meaning of the sentence. Therefore, our error hierarchy is not useful in such cases. In summary, we have been able to use this hierarchy in four different types of tasks: thus we believe it would be sufficiently general to be used in a variety of instructional tasks where the solution can be completely evaluated by its components and their associations.

2.2 Tutorial Dialogues

In our model, error remediation is facilitated through tutorial dialogues. A dialogue is designed for each error type (i.e. each leaf node in the hierarchy), and discusses the domain concept associated with that error. Each dialogue consists of four stages. In the first stage, the dialogue informs the student about the concept that s/he is having difficulty with, and then asks for the justification of the student's action. The purpose of the second stage is to assist the student in understanding why the performed action is incorrect. The third stage prompts the student to specify how to correct the mistake. In the fourth stage, the student can review the domain concept learned.

EERTutor1: You seem to be having some difficulty with regular entities. Can you tell me the general rule to decide whether something is a regular entity?
Student1: *I don't know*
EERTutor2: Let's try a simple thing. How do we model characteristics of an entity?
Student2: *Using Attributes*
EERTutor3 : Great ! Now tell me why modelling *Grade* as an entity is incorrect?
Student3: *I'm not sure*
EERTutor4: Let's see. Check carefully whether you can find any attributes for *Grade*.
{After sometime} Pls. tell me whether you manage to find any attributes for *Grade*
Student4: *No*
EERTutor5: Fine. It doesn't have any attributes. So now can you tell me why modelling *Grade* as an entity is incorrect
Student5: *Because it doesn't have any attributes to describe its characteristics*
EERTutor6: Great! Now tell me the best way to model *Grade*?
Student6: *I need more help*
EERTutor7: Ok. If *Grade* doesn't have attributes, it cannot be an entity. So it should either be a relationship or an attribute. Now tell me the best way to model *Grade?*
Student7: *Attribute*
EERTutor8: Great ! Now use the correct word to fill in the blanks so that you'll remember what you learnt. …….. are used to describe characteristics of an …….. in a ER model
Student8: *attributes, entity*
EERTutor9: Excellent, go ahead and do the changes. Submit the solution when you're done.

Fig. 2: A hypothetical dialogue for the database design tutor

Figure 2 represents a hypothetical dialogue for the database design tutor. Initially, the system identifies the domain concept the student has problems with, and asks the student to explain it (*EERTutor1*). If the student fails to provide the correct answer (*Student1*), s/he will be asked a more specific question that provides a further opportunity to understand the domain concept that is violated (*EERTutor2*). However, if s/he fails to correct the mistake even after going through a series of detailed questions, as the last resort the tutor will provide an explanation on how to the correct the mistake together with a brief description about the domain concept that needs to be learnt (EERTutor4-8). The dialogues consist of simple questions (EERTutor1), fill-in-a-blank (EERTutor8), or true-false questions, to motivate the student to explain. When a certain mistake is repeated, the model informs the

student of its observations (EERTutor1), thereby providing an opportunity to reflect on his/her domain knowledge. As all dialogues facilitate the discussion of errors (EERTutor3), students are given opportunities to reflect on their problem solving procedure, which is another important meta-cognitive skill. Although the prompts are domain-specific, the structure of the dialogues is domain-independent.

2.3 Rules for Adapting Dialogues

Adaptation rules enable individualization of the dialogues, by using the student model to decide on the timing, selection and entry point into the dialogue. Currently there are six rules and they are based on the observations from the study [9]. Some of the rules are discussed here. Rule 1 (dealing with timing of dialogues) checks whether the student made any attempts at the current problem, and has been inactive for a specified period of time (such as 1.5 minutes, the time period we observed in the study). If both these conditions are satisfied, then student's solution is evaluated even though it has not been submitted yet, and a dialogue is initiated to focus on the error most suitable for discussion if multiple errors exist.

Rule 3 addresses the critical issue of selecting a dialogue. Dialogue selection is very important because if it is not effective, it might be difficult for students to systematically develop a comprehensive mental model of the domain. Dialogue selection depends on the student solution and the error hierarchy. The probability of violating a constraint is calculated using the last five submissions on that constraint. For instance, if a constraint is violated twice in the last five submissions, the probability of not knowing it is 0.4. The probabilities of violating individual constraints are then combined to calculate the probability of making an error corresponding to higher-level nodes in the hierarchy. These probabilities are updated each time a student solution is evaluated. Rule 3 finds the error type (e.g. node N1, which is a non-leaf node in the hierarchy) that a student is most likely to make. As the nodes in the hierarchy are ordered from basic domain principles to more complicated ones, the dialogue associated with the left-most leaf node for N1 is chosen as the most suitable dialogue for a set of violated constraints.

Dialogues can be more effective if they are adapted to the student's domain knowledge and reasoning skills. We observed that the tutors tend to discuss the domain concepts relevant for an error if it was done repeatedly [9]. They also tend to state their observations before discussing the domain concept (e.g. "You seem to be having difficulty with regular entities (*EERTutor1* in Figure 2). Rule 4, which deals with the customizing the entry point to the dialogue, is activated when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake is initiated, but the dialogue starts from the problem-independent question (*EERTutor1* in Figure 2). If the error was made less than *n* times, the dialogue will start from the error within the current context (*EERTutor3* in Figure 2). As these rules do not depend on the domain to individualise dialogues, the rules can be used across domains.

Even though this model was developed for constraint-based tutors, it can be used in any ITS providing a problem-solving environment. In such an ITS, a student solution is evaluated and feedback is provided on the errors regardless of the mechanism/methodology used for diagnosis. Therefore, the error hierarchy (the first component of the model) could be developed using the error types of that domain. Tutorial dialogues (the second component of the model) need to be written for each type of error based on the tutorial structure that was discussed in Section 2.2. The third component of the model, rules for adapting dialogues, are domain independent, hence it can be used across domains.

3. Preliminary Study

We conducted an experiment with the ERM-Tutor in April 2006 at the University of Canterbury, which involved volunteers from a database course and experienced tutors. Two types of feedback were provided: typical feedback provided by the system (i.e. hint sequences provided by ERM-Tutor), and dialogues initiated based on the model. The study was conducted as a Wizard-of-Oz study, in which the first author simulated the actions of the model. This additional assistance was given through a chat interface, and will be referred to as interventions hereinafter. Even though the first author used the dialogues from a written script, it was not always possible to use the scripted prompts in the later stages of dialogues. This is due to the student responses not being constrained as in the proposed system. (In the proposed system, the students will be given a list of possible answers from which the correct one can be selected).

Participants interacted with ERM-Tutor in one room, while the first author observed from another room. The participants could initiate interventions through the chat interface or the *More Help* button. Participants were expected to use the system for at least an hour. However, students themselves decided when to end the session. At the end of the session, they filled out a questionnaire. The first phase of the study involved analyzing the logs to investigate the effectiveness of the dialogues. In the second phase, human tutors were asked to judge the appropriateness of interventions by observing recorded sessions. A time line indicating all the interventions was provided to the judges, who indicated whether he/she agrees with the timing and the content of interventions. In the case of a disagreement, the judge was requested to provide justifications.

Ten students and five tutors (acting as judges) participated in the study. The judges were the lecturer and the tutors involved in teaching the course. The average session duration was 59 minutes (sd=15.3). Sometimes the ERM-tutor indicated that the student solution was incorrect even though it was actually correct, due to a coding problem. Such instances were excluded from the analysis. The average number of problems attempted was 11 (sd = 4.6), with 8.4 (sd = 5.2) completed.

From the logs, we identified 65 episodes, each pertaining to a single topic (as in [3]). In addition to facilitating remediation, some episodes focused on helping with the interface (such as moving to the next step), completing the session or helping with technical problems (e.g. web browser not being able to display the page). The number of episodes per session ranged from 1 to 13, with a mean of 6.5 (sd = 4.3). We are mainly interested in 31 episodes in which dialogues were facilitated. Six of these episodes contained a single utterance each, initiated by the wizard. For instance, a tutor utterance that helped a student to understand that multi-valued attributes are not mapped in the first step of the algorithm was "Think about the color attribute". The longest episode consisted of 11 utterances, 6 of which were provided by the model (i.e. the wizard). In the example dialogue (Figure 3), the student is incorrectly applying step 4 to the identifying relationship, while that step should only be applied to regular relationship types. The correct action here is simply to move to the next step. In this situation, the model aims to assist the student to understand that this step is not necessary.

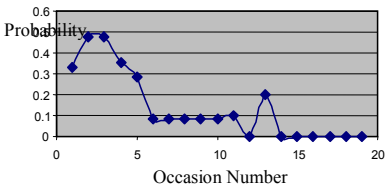


Fig.4. Probability of violating a constraint

In order to investigate whether the dialogues were effective, we analyzed how frequently an error occurred after being discussed in each episode. As the knowledge base in ERM-Tutor is represented as a set of constraints, the errors were recorded as violations of

ermtutor: what do you need to do when you're mapping a 1:N relationship?
coscstudent001: map the n-cardinality table
ermtutor: yes, what is the attribute that needs to be included
coscstudent001: the code from course
ermtutor: yes, good. but can you see this is a special case?
coscstudent001: because section is a weak entity?
ermtutor: yes

Fig. 3. A dialogue from the study

constraints. Thus we analyzed how frequently the constraints that were discussed in the dialogues were violated subsequently.

However, some students were able to correct errors themselves just before the episode started. In another situation, a student indicated that he did not require any assistance (even after a period of inactivity) when prompted. The remaining 15 (48.3%) episodes were included in this analysis. These dialogues involved only seven participants. (The dialogues with the other three students were among the ones excluded.) These dialogues were associated with seven different domain-level constraints. Figure 4 illustrates the learning curve for these constraints. The X-axis represents the occasion number (first, second and so on) when the student violated a constraint discussed in a dialogue subsequently. The Y-axis shows the probability of violating these constraints. The probabilities of violating a constraint on the first and subsequent occasions were averaged over all students. The curve is not smooth due to the small sample size (this analysis involved only seven constraints discussed in 15 dialogues with 7 participants).

However, the learning curve suggests that the probability of subsequently violating a constraint discussed in an episode decreases with occasion number. This indicates that the students seem to learn domain concepts discussed in the episodes, i.e. that the dialogues based on the proposed model did not have a detrimental effect on learning. In order to evaluate whether the remediation facilitated by this model actually enhances learning we need to compare the performance with a control group of students who interact with the system without the dialogues.

In phase 2, five judges analyzed the interventions and indicated whether they agreed with their timing and content. At the beginning of phase 2, they were informed that the goal of the study was to develop a model to facilitate remediation through tutorial dialogues while interacting with a tutoring system. The judges were asked to comment on the appropriateness of the timing and the content of interventions provided through the chat interface. The number of sessions analyzed ranged from 1 to 3 per judge. Due to time constraints, it was not possible to have every session investigated by two judges.

All the episodes were categorized by the rule that initiated them. Five rules were relevant in this study. Rule 1 (described in Section 2.3) was violated only once. Rule 2 (a variation of rule 1) which waits for 1 minute of inactivity after receiving feedback from the system at least once for the current step is violated three times. The tutors agreed with the timing and content of these interventions.

Rule 4 was relevant in 21 episodes and had the highest number of disagreements. Judges disagreed in 7 (33.3%) occasions. The judges disagreed with the content in three situations. For instance, a judge suggested using "*Is there a regular 1:N relationship to map in this problem?*" instead of the first prompt in Figure 3.

One of the issues to be addressed is how to facilitate remediation when nothing needs to be done in a particular step (Figure 3). According to our model, the initial prompt was "*What do you need to do when you're mapping a 1:N relationship?*" which may imply that the student needs to perform an action, even if there is nothing to do. It would be better if the prompt is changed to "*Do you know which type of relationship needs to be mapped in this step*?". The new prompt still discusses a domain concept so it still confines to the dialogue structure discussed in section 2.2.

Some judges preferred earlier interventions than those suggested by the model. The model waits for 3 repeated mistakes before initiating a dialogue. However, it might be effective to intervene after two repeated mistakes, because it is easier to assess what the student is trying to achieve in this particular domain. As the result, the number of times a mistake to be repeated before facilitating remediation may be domain-dependent. Rule 1, which checks for a period of inactivity may also be domain-dependant; however, there was no disagreement on these. Further investigation is needed before the model is changed.

4. Conclusions and Future Work

The research presented in this paper focuses on developing a model for supporting tutorial dialogues for error remediation for both ill- and well-defined tasks. A prototype model was developed based on the findings of a preliminary study using EER-Tutor, an ITS that teaches database design. Database design is an ill-defined task: the final result is defined only in abstract terms and there is no algorithm to find it. In order to investigate the reusability of the proposed model for well-defined tasks, we applied it in three other domains: ER-to-relational mapping, data normalization and fraction addition. We then refined the model based on the findings of this investigation.

This paper focuses on the study which used the model with ERM-Tutor, an ITS developed for the ER-to-relational mapping domain. In addition to the feedback provided by the system, error remediation was facilitated through a chat interface. The interventions through the chat interface were based on the model. Analysis of user logs indicates that students did learn the domain concepts discussed in the dialogues. Human tutors who were asked to analyze the dialogues mostly agreed with the interventions generated by the model. The findings from the reported study are being used to refine the model.

Our next step is to incorporate the model into both EER-Tutor and ERM-Tutor. The enhanced systems will later be evaluated in authentic classroom environments. The goal of these evaluations is to investigate whether the adaptive error remediation supported by the model is more effective in facilitating deep learning than the non-adaptive dialogues, in which two students (with different domain knowledge and reasoning skills) receive the same dialogue when they make the same types of errors in their solutions.

References

[1] Aleven, V., Ogan, A. Popescu, O. Torrey, Cristen, Koedinger, K. R. (2004) Evaluating the effectiveness of a Tutorial Dialogue System for Self-Explanation, Lester, J. C., Vicario, R.M., Papaguacu, F. (eds.)< Proc. Of ITS2004, pp 443-454

[2] Bloom, B. The 2-sigma problem: The search of group instruction as effective as one-to-one tutoring, Educational Researcher 13, 1984 pp.3-16

[3] Chi, M. T. H., S. A. Siler, et al. (2001) Learning from Human Tutoring. *Cognitive Science* 25, 471-533.

[4] Elmasri, R, Navathe, S. Fundamentals of Database Systems. Addison Wesley (2004)

[5] Grasser, A.C., VanLehn, K., Rose, C.P., Jordan P. W., Harter, D. Intelligent Tutoring Systems with Conversational Dialogue AI magazine, 39-51.

[6] Milik, N., Marshall, M., Mitrovic, A. (2006) Teaching Logical Database Design in ERM-Tutor. In: M. Ikeda & K. Ashley (eds). Proc. 8th Int. Conf. on Intelligent Tutoring Systems, pp. 707-709.

[7] Millis, B., Evens, M. , Freedman, R. (2004), Implementing Directed Lines of Reasoning in an Intelligent Tutoring System using the Atlas Planning Environment, International Conference on Information Technology: ITCC 2004, pp. 729-733.

[8] Mitrovic, A., Suraweera, P., Martin, B., Weerasinghe, A. (2004) DB-suite: Experiences with Three Intelligent Web-based database Tutors . Interactive Learning Research, 15(4), 409-432

[9] Weerasinghe, A., Mitrovic, A. (2006) Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based tutors, Aleven V. Ashley, K., Lynch, C. and Pinkwart, N. (eds.), Workshop on ITS for ill-defined domains at ITS2006, pp. 55-64

[10] Weerasinghe, A., Mitrovic, A. (2006) Facilitating Deep Learning through Self-Explanation in an Open-ended Domain, Knowledge-based and Intelligent Tutoring Systems 10(1), 3-19

International Journal of Artificial Intelligence in Education 19(2009) 357-379
IOS Press

Towards Individualized Dialogue Support for Ill-Defined Domains

Amali Weerasinghe, **Antonija Mitrovic**, **Brent Martin**, *Intelligent Computer Tutoring Group*, *University of Canterbury, Christchurch, New Zealand*
Amali.weerasinghe@pg.canterbury.ac.nz, Tanja.mitrovic@canterbury.ac.nz, Brent.martin@canterbury.ac.nz

Abstract. One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. Our goal is to develop a general model for supporting dialogues with menu-based input that could be used in both well- and ill-defined instructional tasks. We have previously studied how human tutors provide additional support to students learning with an existing intelligent tutoring system. On the basis of these findings we developed a model for supporting dialogues, which we present in this paper. We used this model in a Wizard-of-Oz study to provide adaptive support. The results show that students did learn the relevant domain knowledge and that human tutors mostly agreed with the interventions generated from the model.

Keywords. Meta-cognitive skills, tutorial dialogues

INTRODUCTION

Collaborative dialogues between students and tutors are a prominent component of effective tutoring (Graesser, Person & Magliano, 1995). These dialogues provide opportunities for students to reflect on existing knowledge and to construct new knowledge. Such skills are of special importance in ill-defined domains, which are underspecified: they are based on incomplete/imprecise theories, and/or lack prescriptions on how to solve problems.

We start by discussing related work in Section 2, followed by a description of database design, the domain from which we started this project. Then we present a preliminary study of human tutors. This study provided the information for designing our model for supporting adaptive dialogues with menu-based input, which is presented in Section 5. We conducted a Wizard-of-Oz study, using the model to generate dialogue prompts to students in order to evaluate the effectiveness of the model prior to full implementation. This study is presented in Section 6, together with the findings and further improvements to the model. We present the conclusions and the plans for future work in the final section.

RELATED WORK

Some of the dialogue-based tutoring systems that have been developed are Why2-Atlas (Jordan et al., 2006), AutoTutor (Graesser, VanLehn, Rose, Jordan, & Harter, 2001), Why2-AutoTutor (Jackson, Ventura, Chewle, Graesser, & Tutoring Research Group, 2004), CIRCSIM-Tutor (Millis, Evens, &

Freedman, 2004) and Geometry Explanation Tutor (Aleven, Ogan, Popescu, Torrey, & Koedinger, 2004). Why2-Atlas, AutoTutor and Why2-AutoTutor use the dialogues as the main activity to help students learn the domain knowledge. The other systems provide problem-solving environments as the main activity and use tutorial dialogues as a way of remediating errors in student solutions. For example, CIRCSIM-Tutor is a natural language (NL) tutor that helps students solve a set of problems in cardiovascular physiology relating to regulation of blood pressure. The students are engaged in multi-step dialogues based on two experienced human tutors. The dialogue planning is done within the APE framework (Freedman, Rose, Ringenberg, & VanLehn, 2000). Freedman's approach for the tutorial model is a production system that is focused on having a hierarchal view of the dialog. The performance of 50 first-year medical students who interacted with CIRSCIM-Tutor improved significantly.

Geometry Explanation Tutor, an extension of PACT Geometry Tutor (Aleven, Koedinger & Cross, 1999), incorporates natural language understanding (Aleven, Popescu, & Koedinger, 2001) to enhance the learning experience. In Geometry Explanation Tutor, students explain in natural language, and the system evaluates their explanations and provides feedback. The system contains a hierarchy of approximately 200 explanation categories that represent partial or incorrect explanations commonly used by novices (Aleven et. al; 2004). The system parses the student's explanation to generate a semantic representation which is classified according to the hierarchy of explanation categories. The dialogue management system decides the feedback to be presented to the student based on the classification of the student's explanation. An empirical study was carried out to investigate whether self-explanation facilitated through natural language enhances learning better than self-explanation through menu selection. Even though the students who explained in natural language did not learn more than those who explained through menu selection, they did learn better to state explanations.

Atlas-Andes is the product of integrating the Andes physics tutoring system (Gertner & VanLehn 2000) with the Atlas tutorial dialogue system (Freedman et al., 2000). Andes is a model-tracing tutor that presents quantitative physics problems to students. Each problem-solving step entered by a student is highlighted in either red or green to indicate the accuracy of that step. Atlas enhances the learning experience of Andes by leading students through directed lines of reasoning to teach conceptual physics knowledge. When Atlas recognizes an opportunity to encourage deep learning, it initiates a natural language dialogue with the student. The main objective of these dialogues is to facilitate knowledge construction; hence, the dialogues are known as knowledge construction dialogues (KCDs). KCDs provided by Atlas are currently limited to teaching domain principles. An empirical study revealed that the students interacting with Atlas learnt significantly more than students who interacted with ANDES (Rose et al., 2001)

Another tutoring system that engages students in natural language dialogue is Graesser et al.'s AutoTutor (Graesser, Wiemer-Hastings, Wiemer-Hastings, Kreuz, & Tutoring Research Group, 1999). It is used in an introductory course in computer literacy. The system improved the students' learning by 0.5 standard deviation units when compared with a control group of students who read the same chapters from a book. AutoTutor requires students to provide lengthy explanations for the *How*, *Why* and *What-if* type of questions. This approach encourages students to articulate lengthier answers that exhibit deeper reasoning instead of short answers, which may lead to shallow knowledge. A continuous multi-turn dialogue between the tutor and the student takes place throughout the session. The natural language processing components of the tutor are based on Latent semantic analysis (LSA) (Landauer, Foltz & Laham, 1998).

Why2-AutoTutor (Jackson et al., 2004) and Why2-Atlas (Jordan et al., 2006) were developed to facilitate the comparison between the LSA approach used in AutoTutor and the symbolic approach used in Atlas-Andes. Both systems expect the student to write a short essay on a qualitative physics problem. Then the systems analyse the essay and use it as a basis for a tutorial dialogue addressing any misconceptions identified. The systems also provide a critique of the essay and help the student to rewrite it. An empirical study was conducted to test the hypothesis that even when the content is equivalent, students who engage in more interactive forms of instruction learn more. To test this hypothesis, the performance of students who received human tutoring after reading a short text was compared with students who interacted with Why2-Atlas and Why2-AutoTutor. The results revealed that the students learn equally well in all three conditions when the content is at an appropriate level for the student.

The presented systems use different approaches to support tutorial dialogues, depending on the domain and the target student group. CIRCSIM-Tutor, Geometry Explanation Tutor and Atlas-Andes facilitate learning in domains of cardiovascular physiology, Mathematics and Physics. All these domains have well-defined domain theories and the instructional tasks presented to the student are also well-defined (Mitrovic & Weerasinghe, 2009). Therefore, these tutors coach problem-solving in well-defined tasks and use dialogues as a way of remediating student errors. Even though AutoTutor, Why2-AutoTutor and Why2-Atlas support learning in different domains, instructional task supported is eliciting natural language explanations to learn the declarative knowledge. This set of tutors use dialogues as the main method of teaching conceptual knowledge. However a general framework has not been developed to facilitate tutorial dialogues in both well-defined and ill-defined instructional tasks. Our long-term goal is to develop a general model for supporting dialogues via menu-based input that will provide adaptive support to learners across domains. Since we previously incorporated dialogues (Weerasinghe & Mitrovic, 2006a) into a database design tutor (Suraweera & Mitrovic, 2002, 2004), the initial work on this project started with the same tutor. We discuss this domain in the following section.

DATABASE DESIGN: AN ILL-DEFINED TASK

Database design is a process of generating a description of a database using a specific data model. Most database courses teach conceptual database design using the Entity-Relationship (ER) model, a high-level data model originally proposed by Chen (1976). The ER model views the world as consisting of *entities*, and *relationships* between them. The entities may be physical or abstract objects, roles played by people, events, or anything else data should be stored about. Entities are described in terms of their important features (*attributes),* while relationships represent various associations between entities, and also may have attributes. Even though there is a well-defined domain theory (Mitrovic & Weerasinghe, 2009), there is no algorithm to use to derive the ER schema from a given set of requirements. The learner needs to decide on the appropriate constructs to use, such as types of attributes/entities. For example, the learner might be given a problem illustrated in Figure 1 (note that this is a very simple problem). From the problem text, it is obvious that *students* and *groups* are of importance. Therefore, the learner might start by drawing the entities first. Each student has an id, and the learner needs to use his/her world knowledge to realize that ids are unique, and therefore represent that attribute as a key attribute (shown on the diagram as underlined). The number assigned to each group is unique, and therefore it should also be a key attribute. In Figure 1,

the student has made a mistake by showing GROUP as a weak entity, and group number as a partial key. Next, the learner has to think about the relationships between identified entities. In the problem shown in Figure 1, students work in groups, and for each possible association between a student and a group, it is necessary to represent the role. The Role attribute describes the association, and therefore it should be an attribute of the relationship. The student also needs to specify other integrities, such as cardinality ratios (shown as N on the diagram) and participations (shown as single or double lines).

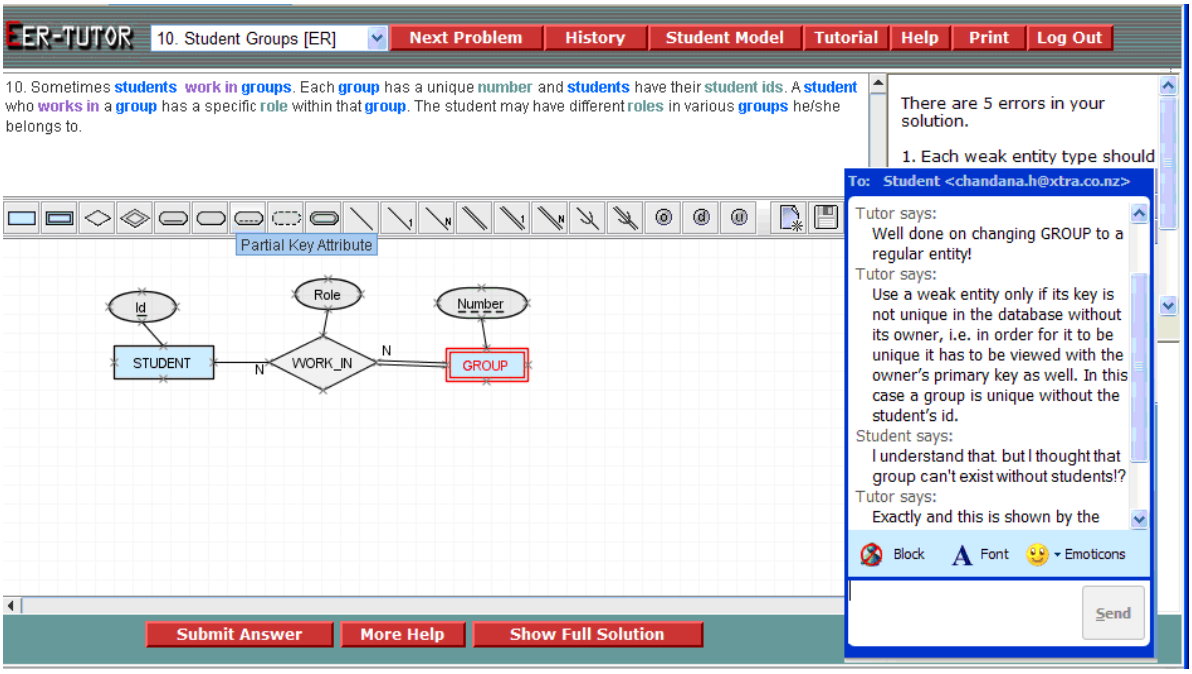


Fig. 1. Interface of the enhanced version of EER-Tutor used in the study.

As can be seen from this simple case, there are many things that the student has to know and think about when designing databases. The student must understand the data model used, including both the basic building blocks available and the integrity constraints specified on them. In real situations, the text of the problem would be much longer, often ambiguous and incomplete. To identify the integrities, the student must be able to reason about the requirements and use his/her own world knowledge to make valid assumptions.

Database design, similar to other design tasks, is an ill-defined task, because the start/goal states and the problem-solving algorithm are underspecified (Reitman, 1964). The start state is usually described in terms of ambiguous and incomplete specifications. The problem spaces are typically huge, and operators for changing states do not exist. The goal state is also not clearly stated, but is rather described in abstract terms. There is no definite test to decide whether the goal has been attained, and consequently, there is no best solution, but rather a family of solutions. Design tasks typically involve huge domain expertise, and large, highly structured solutions.

Although design tasks are underspecified, Goel and Pirolli (1992) identify a set of 12 invariant features of design problem spaces, such as problem structuring, distinct problem-solving phases, modularity, incremental development, control structure, use of artificial symbol systems and others.

Problem structuring is the necessary first phase in design, as the given specifications of a problem are incomplete. Therefore, the designer needs to use additional information that comes from external sources, the designer's experience and existing knowledge, or needs to be deduced from the given specifications. Only when the problem space has been constructed via problem structuring can problem solving commence. The second feature specifies three problem-solving phases: preliminary design, refinement and detail design. Design problem spaces are modular, and designers typically decompose the solution into a large number of sparsely connected modules and develop solutions incrementally. When developing a solution, designers use the limited-commitment mode strategy, which allows one to put any module on hold while working on other modules, and return to them at a later time.

In previous work, we have shown that constraint-based tutors are highly effective in teaching ill-defined tasks such as database design (Suraweera & Mitrovic, 2004) and query definition (Mitrovic & Ohlsson, 1999; Mitrovic et al., 2004). Our tutors compare the student's solution to a pre-specified ideal solution, which captures the semantics of the problem, thus eliminating the need for a problem-solver, which is difficult (or even impossible) to develop for such instructional domains. The constraint-based tutors are capable of identifying alternate correct solutions as constraints check that the student's solution contains all the necessary elements, even though it might be different from the ideal solution specified by the teacher. Even though there can be a family of solutions that are all equally good, the teacher often has a good pedagogical reason for preferring one solution over the others. For example, in database design there are situations where a single higher-order relationship (which has at least three participating entities) or multiple binary relationships (which has two participating entities) can be used. In such a situation, the teacher may prefer one of the solutions. Therefore, it is possible to nominate one "ideal" solution without compromising the quality of the whole ITS, as long as the ITS is capable of identifying other alternative solutions students may come up with as correct.

Goel and Pirolli (1988) argue that design problems by their very nature are not amenable to rule-based solutions. On the other hand, constraints are extremely suitable for representing design solutions: they are declarative, non-directional, and can describe partial or incomplete solutions. A constraint set specifies all conditions that have to be simultaneously satisfied without restricting how they are satisfied. Each constraint tests a particular aspect of the solution, and therefore supports modularity. Incremental development is supported by being able to request feedback on a solution at any time. At the same time, CBM supports the control structure used by the designer (student), as it analyses the current solution looking at many of its aspects in parallel: if a particular part of the solution is incomplete, the student will get feedback about missing constructs. CBM can be used to support all problem-solving phases. Therefore, we believe that CBM can be applied to all design tasks.

HOW DO HUMAN TUTORS SUPPORT ERROR-REMEDIATION?

As the first step towards designing a general model to support tutorial dialogues, we conducted an observational study (Weerasinghe & Mitrovic, 2006b), focusing on how students interacted with EER-Tutor (Zakharov, Mitrovic & Ohlsson, 2005), while getting additional help from a human tutor through a chat interface.

The Experimental Set up

The study was conducted in August 2005 at the University of Canterbury, and involved student volunteers enrolled in an introductory database course and professional tutors. In this discussion we refer to professional tutors as tutors, and to EER-Tutor as the system or the ITS hereinafter. All the tutors had several years of tutoring experience providing assistance on request to students in labs and/or teaching small groups. As EER-Tutor provides a problem-solving environment and complements classroom instruction, the study was scheduled after the relevant learning material was taught in the classroom.

The version of EER-Tutor used in the study was enhanced with a chat interface (Figure 1), so that the tutors could provide one-to-one feedback to students. We wanted to make the bandwidth between the student and the tutor to be very similar to that between the student and the ITS. Therefore, tutors could observe only the students' interactions with the ITS. Participants interacted with the system in one room and the tutors observed their interactions in another room. In the dialogue-enhanced EER-Tutor, dialogues will be used as an additional component to assist problem solving and they would not replace the typical feedback that the system currently provides. Therefore, the participants received both the typical feedback by the EER-Tutor and the additional feedback by the tutors.

We asked the tutors to guide the students towards solutions using appropriate methods like asking questions and so on. However, they were not given any specific instructions on providing assistance. Student participants were not told that a human tutor was involved in the study. They also had the opportunity to initiate intervention through the chat interface or the *More Help* button in the interface.

At the beginning of the study, the participants sat an online pre-test, and then interacted with EER-Tutor. The participants were free to end the session whenever they wanted. All learner interactions were recorded. Although initially we wanted the participants to sit a post-test immediately after the study, it was not possible due to another evaluation study that was conducted simultaneously. Therefore, the post-test was administered later on. All participants were asked to fill out a questionnaire at the end of the session to understand their perceptions about the system and interventions through the chat interface. At the end of each session, the tutors were also interviewed to understand their views on the tutoring experience. Two tests of comparable difficulty were scheduled to be used as pre and post-tests. Each test contained 7 questions. Each correct answer scored one mark and the total score was given out of seven.

We analysed the recordings to investigate how students were prompted by different tutors and which interactions triggered these prompts. In the second phase, whenever possible, we discussed the recordings with the tutors to clarify how they decided on the timing and the level of feedback provided through the chat interface. This experimental set up varies from previous studies of tutorial dialogue in a number of ways. First, the human tutor in this study provided support in addition to the feedback given by the system. The tutors also responded to learners' questions. This contrasts with those studies of Chi, Siler, Jeong, Yamauchi, and Hausmann (2001) and Graesser, Person and Magliano (1995), in which the tutor is expected to lead the dialogue through a series of questions. Second, the learner interacted both with the system and the tutor. Although Merrill, Reiser, Raney, and Trafton (1992) have studied tutorial dialogues in the context of problem-solving, the tutor was the only source of feedback for the student as s/he solved problems on paper. Finally, the tutors in our study needed to decide not only how to guide the student but also when. This differs significantly from the study in which the tutors analysed recorded interactions of students to perform motivation diagnosis (De

Vicente, & Pain, 2002). However this setup is similar to the study conducted by Rose and her colleagues (Rose, Fumer, Aleven, Robinson, & Wu, 2006).

Observations

Seven students and four professional tutors participated in the study, with at most two students per tutor. The mean on the pretest was 75.5% (sd=17.9), which was higher than the performance of the whole class (mean=58.1, sd=23.5). We expected this, as the participants were self-selected. Still the range of background knowledge was sufficiently large (ranging from 57% to 100%). The posttest was available online after a pre-specified date. Only two students completed the post-test, hence it is not possible to compare the effect the learner interactions had on performance. The average duration of the sessions was 85 minutes (sd=20). The average number of problems attempted was 11 (sd = 5) and all participants completed all attempted problems. We discuss observations in two different categories: (1) type of feedback provided in the interventions and (2) timing of interventions.

Type of Feedback Provided

We analysed the interactions between the tutors and the students in order to identify episodes, each pertaining to a single topic (Chi et al., 2001). There were a total of 69 episodes. In addition to discussing the current problem state, some episodes focused on helping with the interface (such as labeling constructs), motivating and praising the student, suggesting trying a more challenging problem, completing the session or helping with technical problems (e.g., web browser suddenly closing). The number of episodes initiated by a tutor per session ranged between 4 and 20. Surprisingly, these 20 episodes occurred in a session of 1.5 hour duration which is not the longest session (the longest session lasted approximately 2 hours). In the session which included 4 episodes, the first intervention occurred only in the 19th problem (the student completed 22 problems).

We are mainly interested in 37 episodes that discussed the current problem state or the relevant domain concepts. The following statistics were calculated using these 37 episodes. The average number of such episodes per tutor was 9.25. Five episodes contained a single utterance each, which was initiated by the tutor. For instance, a tutor utterance that occurred just after the completion of a problem was "*Remember that the participation for weak entity is always total.*" The longest episode consisted of 9 utterances of which 4 were by the tutor. The student made more utterances than the tutor in only 2 episodes. Furthermore, only 2 episodes were student-initiated. This indicates that the tutor is more likely to be active in the interventions.

An example is presented in Figure 2, which occurred while a student was solving the problem in Figure 1. In this dialogue, the student was able to identify and repair the misconception he had with weak entities and total participation.

The highest number of dialogues in a session was 7, while the lowest was 2. As can be expected, the highest number of dialogues occurred in the longest session.

From the collected data, we identified three techniques used by human tutors. Tutors were rephrasing feedback from the ITS, providing problem-independent explanations and stating the tutor's observations before starting to discuss the problem state. The tutors rephrased feedback to enable the student to understand their own mistake. For example, the tutor prompted "*Does AUTHOR need to be an entity?*" or "*The cardinalities of BORROWED_FROM needed fixing.*" Rephrasing feedback may have been effective because most students realised that the additional feedback was provided by a

human observing their problem-solving process. If our model is to repeat the same kind of prompting, it is difficult to ascertain whether it will have the same effect (Lepper, Woolverton, Mumme, & Gurtner, 1993). The second technique was to discuss the current problem state and then provide a problem-independent explanation. Figure 2 represents an example. These explanations provided an opportunity for the student to repair his mental model of the domain and generated further conversation. The third technique was to state the tutor's observations before starting to discuss the problem state. For example, tutor started the dialogue by saying "*You seem to be having a few problems with relationships. Think about this. Can a student be enrolled in a course without involving a department?*"

Tutor: Well done on changing GROUP to a regular entity!
Tutor: Use a weak entity only if its key is not unique in the database without its owner,
i.e. in order for it to be unique it has to be viewed with the owner's key as well.
In this case a group is unique without the student's id.
Student: I understand that, but I thought that group can't exist without students!?
Tutor: Exactly, and this is shown by the total participation in the relationship.
Student: I have learned something today

Fig. 2. Example of a self-explanation episode.

As the knowledge base in EER-Tutor is represented as a set of constraints, the errors were recorded as violations of constraints (Zakharov, Mitrovic & Ohlsson, 2005). We analysed how frequently constraints were violated after related errors were discussed using dialogues, to see whether tutor interventions helped students to improve their knowledge. If these constraints represent psychologically appropriate units of knowledge, then learning should follow a smooth curve when plotted in terms of constraints (Anderson, 1993). To see whether tutorial dialogues are useful, the participants' logs were analysed, and each problem-state after a tutor intervention in which a constraint was relevant was identified. These identified occasions are referred to as *occasions of application*. Each constraint relevance occasion was ranked 1 to *n*. For each occasion we recorded whether a relevant constraint was satisfied or violated. We then calculated the probability of violating a constraint on the first occasion of application, the second occasion and so on, for each participant. The probabilities were then averaged across all participants and plotted as a function of the number of occasions when a constraint was relevant (Figure 3). At the first occasion, all participants had some relevant constraints, whereas only two participants had a constraint relevant at n = 17. This indicates that the number of constraints that were relevant decreases as occasion number increases.

As can be seen from Fig. 3.a, there is an outlier, increasing the probability of violating a constraint in the 4^{th} and the 5^{th} occasions. This is due to a single student violating the constraint dealing with total participation of entities. For this student, the tutor provided a problem-independent explanation on total participation to help him identify and repair the misconception he had with weak entities and total participation (Figure 2). The explanation was not related to a problem state later on, as the discussion was a follow-up from another error related to weak entities. This may have been a reason for the subsequent violations of this constraint.

Figure 3.b shows the learning curve with the outlier removed. The probability of 0.22 for violating a constraint at its first occasion of application decreased to 0.02 at its eighth occasion of application, displaying a 90.9% decrease in the probability. The results of the mastery of constraints reveal that students seem to learn ER modelling concepts that were discussed by the tutors.

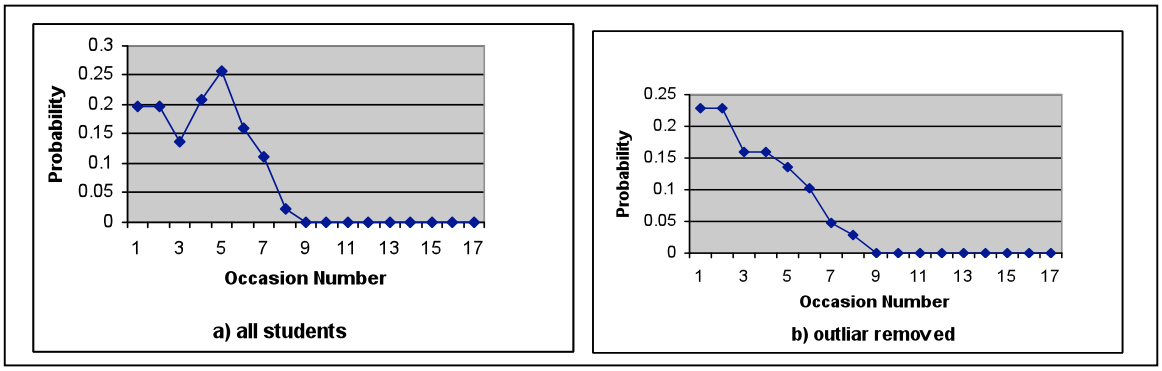


Fig. 3. Probability of violating a constraint.

Twenty-eight different constraints were discussed in the dialogues. Three students did not violate any constraints in subsequent occasions after the tutor interventions. These students were tutored by three different tutors who followed techniques like rephrasing feedback, providing problem-independent explanations and stating tutor's observations at the beginning of the discussion. This suggests that all these techniques have been effective in helping the students learn domain concepts.

Timing of Interventions

The original version of EER-TUTOR provides feedback on demand, that is, only when the student submits the solution. The tutors in this study also provided delayed feedback, which was well-received by the participants. Delayed feedback also provided an opportunity for students to correct the mistakes themselves. There were few instances where the student made a mistake and corrected it after referring the problem text again. For example, one of the problems required students to model CAR as an entity and *Colour* as a multi-valued attribute of CAR. The student modelled *Colour* as a simple attribute and then changed it to multi-valued as the last sentence in the problem text indicated that a car can have many colours. In such a situation, immediate feedback would not have been welcomed by the student as he may have felt the intervention to be intrusive.

The important issue with delayed feedback is how the tutors decided that the students needed help. In our study, tutors provided help when the student (i) made the same type of mistake repeatedly, (ii) asked for more help using the *More Help* button, (iii) was inactive for some time, (iv) reacted to feedback, or (v) asked a problem-specific question through the chat interface.

PROTOTYPE OF THE MODEL

The goal of our project is to provide adaptive dialogue support, which means that the model should determine when to initiate dialogues, what to discuss and how to obtain explanations from learners. We designed the model on the basis of our previous work (Weerasinghe & Mitrovic, 2006a) and the findings from the study of human tutors presented in the previous section. The model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level an error type is associated with one or

more violated constraints, which form leaves of the hierarchy. The error types are then grouped into higher level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue, learners are able to provide answers by selecting the correct option from a list. Each component is now described in detail.

Error Hierarchy

In previous work, we developed a hierarchy of errors students make in the Entity-Relationship (ER) domain (Weerasinghe & Mitrovic, 2006b), which categorizes errors as being syntactic or semantic in nature. A high-level view of the hierarchy is given in Figure 4, showing the top three levels only. In constraint-based ITSs, violated constraints indicate errors in a student solution: violated constraints for each type of error, therefore, form the leaves of the hierarchy. Syntax errors are generally simple, each requiring only one feedback message to be given to the student rather than initiating a dialogue; for that reason, every syntactic error tends to correspond to a particular constraint being violated. Twelve constraints are associated with syntax errors for the ER domain; for example, constraint 7 is violated when two entities are directly connected to each other, and forms a single error type. In contrast, the hierarchy for semantic errors is deeper because constraints are often related by some high-level concept. For semantic errors, error types are typically further divided into sub-errors.

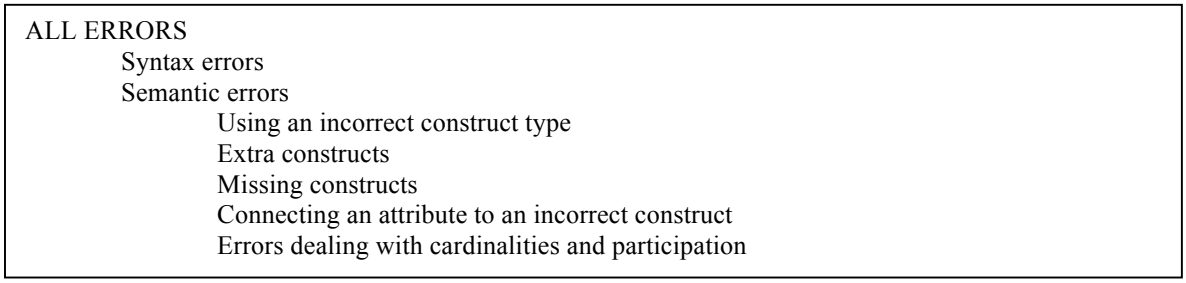


Fig. 4. Overall view of the error hierarchy.

The original error hierarchy (Weerasinghe & Mitrovic, 2006b) was developed for the ER modeling domain, so we were interested whether it could be reused in other domains. With that goal, we tried to fit the errors from a different domain, logical database design, into this structure. This domain involves mapping high-level, conceptual ER schemas to relational schemas using the 7-step mapping algorithm (Elmasri & Navathe, 2007). The task is well-defined due to the deterministic algorithm used. However, both domains (ER modeling and ER-to-relational mapping) involve mapping as the major activity. Due to this similarity, we decided to explore additional domains of different nature, such as data normalization and fraction addition. Data normalization is the process of refining a relational database in order to ensure that all relations are of high quality (Elmasri & Navathe, 2007).

During this investigation, we identified situations when it was not enough to present a single feedback message for some violated syntax constraints: a dialogue was required. Therefore, we

modified the structure of the error hierarchy to divide all error types into two main categories: *Basic Syntax Errors* and *Errors dealing with the main problem-solving activity* (Figure 5). Under the new node *Basic Syntax errors*, we included simple syntax errors such as checking whether the student has filled the required fields, the components used to fill the required fields are valid and so on. Hence it is sufficient to discuss such errors using a single message. The other category requires a dialogue to be conducted. We also decided to change the names of some nodes in the highest level to make the hierarchy more general. For instance, *Extra constructs* was renamed to be *Extra solution components, Missing constructs* to *Missing solution components* and so on.

Another observation was that different clusters of syntax errors were needed in these newly examined domains, as opposed to the flat structure of the syntax error sub-hierarchy in Figure 4. For example, in data normalization, several constraints check whether the student is using valid attribute names in different steps of the algorithm, all of which can be categorized into a single node (*Check Validity of attributes),* specified as a child node of *Basic Syntax Errors.*

Another refinement required was to make the two domain-specific nodes *Connecting an attribute to an incorrect construct* and *Errors dealing with cardinalities and participation* more general so that the overall hierarchy can be used across domains. As these two nodes deal with associations between solution components, it is appropriate to have a new node *Associations* (Figure 5). This new node has different domain-specific child nodes.

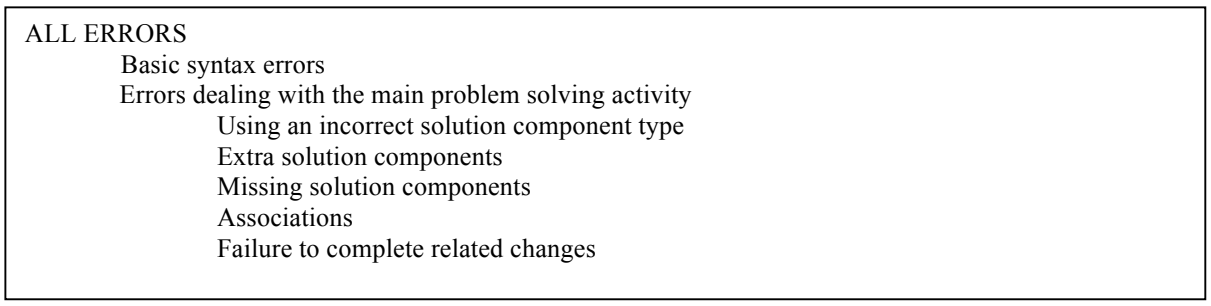


Fig. 5. Overall view of the refined error hierarchy.

A final refinement was made based on an observation from the study of human tutors reported in the previous section: some students seemed to be reacting to feedback on errors by making suggested changes without reflecting on other modifications that also needed to be carried out. As an illustration, in ER modeling if a regular entity with a key attribute is changed to a weak entity, a partial key should be specified instead of the key attribute. This may have led to frustration due to the number of attempts that the student had to go through to arrive at the correct solution. A new node *Failure to complete related changes* was added to the existing error hierarchy, which reminds the student to check whether other changes are necessary (Figure 5). In such cases, the student will be prompted to reflect on other related changes before submitting the solution.

Figure 5 only shows the three highest levels of the refined error hierarchy, as the lower levels deal with domain-specific concepts. We also tested the refined error hierarchy in the domain of fraction addition. Even though this domain is very simple, it is quite different from the domains that we have investigated previously. All the error types in the fractions domain could be specified using the error hierarchy.

Using an incorrect solution component type
Using a completely different type of solution component
Using an entity to represent another type of solution component
Using an entity to represent a relationship
(27 or 28)
Using an entity to represent an attribute
(202 or 203 or 204 or 205 or 206 or 202-1or 205-1or 206-1)
Using another type of solution component to represent an entity
Using a relationship to represent an entity
(13_1or 14_1)
Using an attribute to represent an entity
(13_2 or 14_2)
65-5
65-6
Other representations
Using a relationship to represent an attribute
207 or 208 or 209 or 210 or 211_A or 207_1 or 210-1 or 211-B
Using an attribute to represent a relationship
27_2 or 28-2

Using a different variation of the correct solution component
Entity
Using a regular entity to represent a weak entity
(14)
(67-2)
Using a weak entity to represent a regular entity
(13)
Relationship
Using a regular relationship to represent an identifying relationship
(28_1)
Using an identifying relationship to represent a regular relationship
(27-1)
Attribute
Using a different type of attribute
(54 or 55 or 56 or 57 or 58 or 59 or 54_1 or 57_1 or 58_1 or 59-1)
65-2

Fig. 6. Detailed view of the node *Using an incorrect solution component type* for Database Design.

Figure 6 represents the detailed view of the node *Using an incorrect solution component type* for database design. This node is further divided into two categories: *Using a completely different type of solution component* and *Using a different variation of the correct solution component*. When a relationship in the ideal solution (IS) is represented as an entity in the student's solution (SS), this

error is categorized under the first sub node (*Using a completely different solution component*). For instance, constraint 27 is violated when an entity is used to model a regular relationship in the IS. On the other hand, constraint 28 is violated when an entity is used to represent an identifying relationship. The second sub node (*Using a different variation of the correct solution component*) deals with more subtle errors such as a weak entity being represented as a regular entity in the SS or a regular relationship being represented as an identifying relationship. The content of the error hierarchy (the number of levels and the nodes) depends on the domain. Figure 7 represents the detailed view of the node *Using an incorrect solution component type* for the fraction addition domain. As can be expected, the subtree for this node for fraction addition is much simpler than that of the ER modeling. The node *Associations* forms the largest tree with 8 levels for the ER modelling whereas the node *Extra solution components* forms the smallest with 3 levels.

Using an incorrect solution component type
Using an incorrect least common denominator (LCD)
Higher multiple of the correct LCD
10-gse
Incorrect LCD
0-gse
Using an incorrect denominator
Fraction1
5-gse
Fraction 2
4-gse
Sum
2-gse
Using an incorrect whole number
7-gse

Fig. 7. Detailed view of the refined node *Using an incorrect solution component type* for fraction addition.

The common feature in all these tasks is that the syntactic and semantic accuracy of a solution can be completely evaluated by the components of the solution and its associations. However, there are exceptions. For instance, in reading and comprehension, where learners are asked to answer questions based on a paragraph, the accuracy of an answer cannot be evaluated by checking only for the correct words according to the grammatical rules. We also need to understand the implicit semantic meaning of the sentence. Therefore, our error hierarchy is not useful in such cases. In summary, we have been able to use this hierarchy in four different types of tasks: thus we believe it would be sufficiently general to be used for different types of instructional tasks only when the solution can be completely evaluated by the components of the solution and its associations.

If the error hierarchy is developed for an existing constraint base, then the hierarchy can also be used to understand/verify how comprehensive the constraint base is. When the hierarchy was developed for the fraction addition domain, several missing constraints were identified. For example, two new constraints were needed when a student has correctly converted the denominator but failed to do the same for the numerator of the given fractions fraction1 and fraction2. When we were exploring the data normalization domain, several constraints that needed to be made more specific were

identified. For example, when simplifying functional dependencies (fds) one of the initial checks was to compare the number of fds of the IS with that of the SS. This constraint can be violated either by having too many or too few fds in the SS. As a result, this constraint can be categorized under both *Missing solution components* or *Extra solution components*, leading to two different dialogues. Therefore, this constraint needs to be made more specific so that it deals with only one type of error.

Tutorial Dialogues

In our model, error remediation is facilitated through tutorial dialogues. A dialogue is designed for each error type (i.e., each leaf node in the hierarchy). As the domain model of constraint-based tutors is represented as a set of constraints, violations of constraints indicate errors in a student solution. In other words, an error in a student solution indicates the domain concept that the student has difficulty with. Each dialogue therefore discusses the domain concept associated with that error.

Each dialogue consists of four stages. In the first stage, the dialogue informs the student about the concept that s/he is having difficulty with, and then asks for the justification of the student's action. The purpose of the second stage is to assist the student in understanding why the performed action is incorrect. The third stage prompts the student to specify how to correct the mistake. In the fourth stage, the student can review the domain concept learned.

Different stages of the dialogue address different aspects of a domain concept. The first stage facilitates the understanding of the concept whereas the final one provides an opportunity to reinforce it. The second stage helps the students to understand how s/he has applied the concept incorrectly to the current context. The third stage provides an opportunity to apply the domain concept correctly.

Dialogues are developed as tree structures and authored manually. The top-level prompt is problem-independent. It consists of an observation ("You seem to be having some difficulty with regular entities" - *EERTutor1* in Figure 8) and an opportunity to discuss the reasoning behind the student's problem solving step. Starting the discussion by stating the domain concept that the student has difficulty with is one of the strategies used by the observational study described previously. Starting the discussion with such an observation is appropriate as the top-level prompt is used only after the student has made the same error three times repeatedly. If the student's response is incorrect, the ITS guides the student through a remediation recipe. Remediation recipes can take different forms. In simple cases, a brief description of the domain concept is presented to help the student to acquire missing information (*EERTutor2* in Figure 8). In other cases, a more specific prompt is used to guide the student to provide the correct answer. After going through the remediation recipe, the ITS will move to the next stage. A correct response at each stage will activate the next level prompt of the dialogue.

A hypothetical dialogue for database design is given in Figure 8,[1] shown as a linear sequence instead of the full tree due to space restrictions. It is initiated when a student uses a regular entity in a situation when an attribute should be used. This error is categorized under the node *Using an incorrect solution component type*. Initially, the system identifies the domain concept the student has problems with, and asks the student to explain it (*EERTutor1*). If the student cannot answer (*Student1*), s/he will be given a brief description that provides a further opportunity to understand that domain concept (*EERTutor2*). The dialogues consist of simple questions (*EERTutor1*), fill-in-the-blank (*EERTutor7*), or true-false questions, to motivate the student to explain. As all dialogues discuss errors (*EERTutor2*),

[1] The complete dialogue is available at http://www.ictg.canterbury.ac.nz/dialogue-enhanced-tutors.

students are given opportunities to reflect on their problem solving procedure, which is another important meta-cognitive skill.

EERTutor1: You seem to be having some difficulty with regular entities. Let's look at regular entities in detail. Can you tell me the general rule to decide whether something is a regular entity?
Student1: *I don't know*
EERTutor2: Ok ! Please remember that if something has its won attributes then we model it as an entity. Now can you tell me why modeling *Grade* as an entity is incorrect?
Student2: *I'm not sure*
EERTutor3: Let's see. Check carefully whether you can find any attributes for *Grade*. Please press <Continue> when you are ready.
{After <Continue> is pressed} Please tell me whether you can find any attributes for *Grade*
Student3: *No*
EERTutor4: Fine. It doesn't have any attributes. So now can you tell me why modelling *Grade* as an entity is incorrect
Student4: *Because it doesn't have any attributes to describe its characteristics*
EERTutor5: Great! Now tell me the best way to model *Grade*?
Student5: *I need more help*
EERTutor6: Ok. If *Grade* doesn't have attributes, it cannot be an entity. So it should either be a relationship or an attribute. Now tell me the best way to model *Grade?*
Student6: *Attribute*
EERTutor7: Great ! Now use the correct word to fill in the blanks so that you'll remember what you learnt.
…….. are used to describe characteristics of an …….. in a ER model
Student7: *attributes, entity*
EERTutor8: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 8. A hypothetical dialogue in Database Design.

Figure 9 represents a hypothetical dialogue in fraction addition domain. It is initiated when the student is having difficulty calculating the least common denominator (LCD) for adding two fractions. The first prompt *MathTutor1* intends to tell the student about the difficulty that he/she is having and assist him/her to understand the general concept of LCD. The prompt *MathTutor2* provides an opportunity to understand why the calculated LCD is incorrect. The third stage in which student is given an opportunity to understand how to correct the mistake consists of *MathTutor3- MathTutor9*. This stage is much longer and goes through several turns for this particular error compared to the dialogue in Figure 8. The dialogue finishes with a question for reflection (*MathTutor10).* As can be expected, the number of prompts for the third stage can be different depending on the error type and the domain. Although the prompts are domain-specific, the structure of the dialogues is domain-independent.

Rules for Adapting Dialogues

Adaptation rules enable individualization of the dialogues by using the student model to decide on the timing, selection and entry point into the dialogue. Table 1 presents the current set of rules, which are based on the observations from the study reported in Section 3. Some of the rules are discussed here. As these rules do not depend on domain-specific details to individualise dialogues, the rules can be used across domains.

Rule 1 (dealing with timing of dialogues after a period of inactivity) checks whether the student made any attempts at the current problem, and has been inactive for a specified period of time (such as 10 minutes, the time period we observed in the study). If both conditions are satisfied, then student's

solution is evaluated even though it has not been submitted yet, and a dialogue is initiated to focus on the error most suitable for discussion if multiple errors exist.

MathTutor1: You seem to be having some difficulty calculating the least common denominator (LCD). Let's look at LCD in detail. Can you tell me why we need the LCD?
Student1: *I don't know*
Ok ! Please remember that LCD is used to convert denominators of the fractions to be added to a common denominator. Now tell me why having 12 as the LCD of 6 and 8 is incorrect?
Student2: *I'm not sure*
MathTutor3: Let's see …Check carefully whether there are any common factors for 6 and 8? Please press <Continue> when you are ready.
{After <Continue> is pressed} Please tell me whether you manage to find any common factors for 6 and 8?
Student3: Yes
MathTutor4: What are the common factors?
Student4*: It's 2.*
MathTutor5: Good job! Can you also check whether 8 is a higher multiple of 6?
Student5: *No*
MathTutor6: Good, then do you know why having 12 as LCD is incorrect?
Student6*: because you can't convert denominator of fraction2 (i.e. 8) to 12*
MathTutor7: Great! Now tell me the correct LCD in this case?
Student7: *I need more help*
MathTutor8: Ok. When you have two numbers with common factors but one is not the higher multiple of the other, step1 is to find the prime factors for each integer. Try to find the prime factors for each integer? Please press <Continue> when you are ready.
{After <Continue> is pressed}, pls. tell me the prime factors that you have found?
Student8: 6= 2 *3; 8 = 2^3
MathTutor9: Great! Now the next step is to find the LCD. It is the product of multiplying the highest power in each prime factor category together. Can you tell me the correct LCD?
Student9: *24*
MathTutor10 : Well done! Now use the correct word to fill in the blanks so that you'll remember what you learnt.
…….. is used to convert the …….. of the fractions to be added to a common ……..
Student10: *LCD, denominators, denominator*
MathTutor11: Excellent, Go ahead and do the changes. Submit the solution when you're done.

Fig. 9. A hypothetical dialogue in the Fraction Addition domain.

Rule 3 addresses the critical issue of selecting a dialogue. Dialogue selection is very important because if it is not effective, it might be difficult for students to systematically develop a comprehensive mental model of the domain. Dialogue selection depends on the student solution and the error hierarchy. When multiple errors exist in a student solution, we would like to focus on the error that the student is most likely to make in subsequent attempts. This is based on the probability of violating a constraint in the last five submissions of the student solution. The probabilities of violating individual constraints are then combined to calculate the probability of making an error corresponding to higher-level nodes in the error hierarchy. These probabilities are updated each time a student solution is evaluated. Rule 3 finds the error type (e.g., node N1, which is a non-leaf node in the hierarchy) that a student is most likely to make in future attempts. As the nodes are ordered from basic domain principles to more complicated ones, the dialogue associated with the left-most leaf node for N1 is chosen as the most suitable dialogue for a set of violated constraints. However, there is one exception: if a sub node which comes under *Missing solution components* has the highest probability of making a mistake, that indicates the SS has a considerable number of missing solution components. As it is more effective from a pedagogical point of view to discuss what the student has already

attempted rather than the ones he has not completed, we will be looking for the node with the second highest probablity. The errors that are categorized under *Missing solution components* are considered only when the errors that come under other nodes have been corrected.

Dialogues can be more effective if they are adapted to the student's domain knowledge and reasoning skills. We observed that the tutors tend to discuss the domain concepts relevant for an error if it was done repeatedly. They also tend to state their observations before discussing the domain concept (e.g., "You seem to be having difficulty with regular entities (*EERTutor1* in Figure 3)). Rule 4, which deals with customizing the entry point to the dialogue, is activated when the same error is made in the last *n* attempts. In that case, a dialogue corresponding to the mistake is initiated, but the dialogue starts from the problem-independent question (*EERTutor1* in Figure 3). If the error was made less than *n* attempts, then the dialogue will start from the error within the current context (*EERTutor3* in Figure 3).

Even though the dialogues were intended to facilitate error remediation, they might be felt as a burden to some natural explainers. Hence, we do not expect the students to go through the entire dialogue when their solution is erroneous. Rule 5 monitors when they have answered the dialogue prompt correctly indicating that they may have understood their mistake. In such a case, the students are encouraged to resume problem solving.

The short-term student model in constraint-based tutors consists of lists of satisfied/violated constraints for the student's solution, while the long-term model records constraint histories. We will extend both types of student models to additionally record details of the student's reasoning skills in terms of types of errors made (i.e., rules that were initiated) and the level of prompting the student needed to correct errors for each domain-level constraint. This additional information can be used to identify whether a student has improved his/her reasoning skills (ideally they need less prompting to understand their mistakes), and the dialogues that are most effective.

Even though this model was developed for constraint-based tutors, it can be used in any ITS providing a problem-solving environment. In such an ITS, a student solution is evaluated and feedback is provided on the errors regardless of the mechanism/methodology used for diagnosis. Therefore, the error hierarchy (the first component of the model) could be developed using the error types of that domain. Tutorial dialogues (the second component of the model) need to be written for each type of error based on the tutorial structure that was discussed in section 2.2. The third component of the model, rules for adapting dialogues, are domain independent, hence it can be used across domains.

PRELIMINARY STUDY OF THE PROPOSED MODEL

We conducted an experiment with the ERM-Tutor (Milik, Marshall, & Mitrovic, 2006) in April 2006 at the University of Canterbury, which involved student volunteers and experienced tutors. Two types of feedback were provided: typical feedback provided by the system, and dialogues initiated by the model. The study was conducted as a Wizard-of-Oz study, in which the first author of this paper simulated the actions of the model. This additional assistance was given through a chat interface (see Figure 10), and will be referred to as interventions hereinafter. The first author used the dialogues from a written script. However, it was not always possible to use the scripted prompts in the later stages of dialogues because the student answers were not constrained as in the proposed system. (In

the proposed system, the students will be given a list of possible answers from which the correct one can be selected.).

Table 1
Adaptation Rules

Rule Identifier	Adaptation Rule
1	IF SS has not been changed for the last 10 minutes and has not been evaluated at all, THEN evaluate SS and start the dialogue.
2	IF SS has not being changed for the last 5 minutes and has been evaluated previously THEN evaluate SS and initiate dialogue
3	**Selecting the dialogue** IF SS is incorrect THEN select the left-most dialogue with the highest probability of making a mistake and display the message (about the observation that the student is having problems with a specific sub area) .
4	**Selecting the entry point of the chosen dialogue** IF the same mistake is repeated 3 times (this includes errors with the pre-test) within the same session, THEN start the explanation from problem-independent prompt of the chosen dialogue ELSE start the explanation from problem-dependent prompt of the chosen dialogue
5	**Moving to the next level of the dialogue** IF the student response for the current prompt is incorrect THEN move to the next level prompt ELSE encourage student to resume problem solving
6	IF the student has changed a problem without completing it (for well-defined tasks, if the problem is changed after completing an intermediate step) THEN evaluate SS, inform the student that he has not completed the problem, and ask the student if s/he wants to change the problem or wants help.
7	IF the student has changed 3 problems without completing them consecutively (for well-defined tasks, if the problem is changed after completing an intermediate step) THEN identify whether they stop when they face the difficulty in a certain sub-area, and state the observation

Participants interacted with ERM-Tutor in one room, while the first author observed from another room. The participants could initiate interventions through the chat interface or the *More Help* button. Participants were expected to use the system for at least an hour. However, students themselves decided when to end the session. At the end of the session, they filled out a questionnaire. The first phase of the study involved analyzing the logs to investigate the effectiveness of the dialogues. In the second phase, human tutors were asked to judge the appropriateness of interventions by observing recorded sessions. A time line indicating all the interventions was provided to the judges, who

indicated whether he/she agrees with the timing and the content of interventions. In the case of a disagreement, the judge was requested to provide justifications.

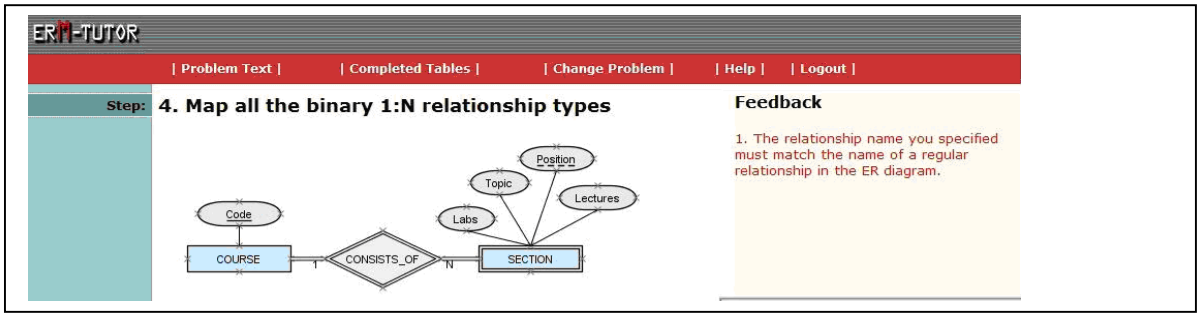


Fig. 10. The screenshot of the version of ERM-Tutor used in the preliminary study.

Ten students and five tutors (acting as judges) participated in the study. All students were enrolled in an introductory database course at the University of Canterbury. The judges were the lecturer and the tutors involved in teaching this course. The average session duration was 59 minutes (sd=15.3). In some cases, the ERM-tutor indicated that the student solution was incorrect even though it was actually correct, due to bugs in the system. Such instances were excluded from the analysis. The average number of problems attempted was 11 (sd = 4.6), with 8.4 (sd = 5.2) problems completed on average.

From the logs, we identified 65 episodes. An episode is considered to be a multi-turn dialogue pertaining to a single topic (Chi, 2000). In addition to facilitating remediation, some episodes focused on helping with the interface (such as moving to the next step), completing the session or helping with technical problems (e.g., web browser not being able to display the page). The number of episodes per session ranged from 1 to 13, with the mean of 6.5 (sd = 4.3). We are mainly interested in 31 episodes that facilitated error remediation. Six of these episodes contained a single utterance each, initiated by the wizard. For instance, a tutor utterance that helped a student to understand that multi-valued attributes are not mapped in the first step of the algorithm was "think about the color attribute." The longest episode consisted of 11 utterances, 6 of which were provided by the model (i.e., the wizard). An example is given in Figure 11. In this dialogue, the student is incorrectly applying step 4 (mapping 1:N relationships) to the identifying relationship, while that step should only be applied to regular relationship types. The correct action here is simply to move to the next step. In this situation, the model aims to assist the student to understand that this step is not necessary.

ermtutor: What do you need to do when you're mapping a 1:N relationship?
student001: Map the n-cardinality table
ermtutor: Yes, what is the attribute that needs to be included
student001: The code from course
ermtutor: Yes, good. But can you see this is a special case?
student001: Because section is a weak entity?
ermtutor: Yes

Fig. 11. A dialogue from the study.

In order to investigate whether the dialogues were effective, we analyzed how frequently an error occurred after being discussed in each episode. As the knowledge base in ERM-Tutor is represented as a set of constraints, the errors were recorded as violations of constraints. Thus we analyzed how frequently the constraints that were discussed in the dialogues were violated subsequently. However, some students were able to correct the mistake themselves just before the episode started. In another situation, a student indicated that he did not require any assistance (even after a period of inactivity) when prompted. Also, some violated constraints were not consistent with the actual mistake in the student solution due to a coding problem. It was not possible to specify a constraint for such episodes. Thus, those episodes were excluded for the analysis. The remaining 15 (48.3%) episodes were included in this analysis.

The selected dialogues involved only seven participants. (The dialogues with the other three students were among the ones excluded.) These dialogues were associated with seven different domain-level constraints. Figure 12 illustrates the learning curve for these constraints. The curve is not smooth due to the small sample size (i.e., this analysis involved only seven constraints discussed in 15 episodes with 7 participants), but it does suggests that the probability of subsequently violating a constraint discussed in an episode decreases with occasion number. This indicates that the students seem to learn domain concepts discussed in the episodes, that is, that the dialogues based on the proposed model did not prevent learning. In order to evaluate whether the dialogues facilitated by this model actually enhance learning, we need to compare the performance with a control group of students who interact with the system without the dialogues.

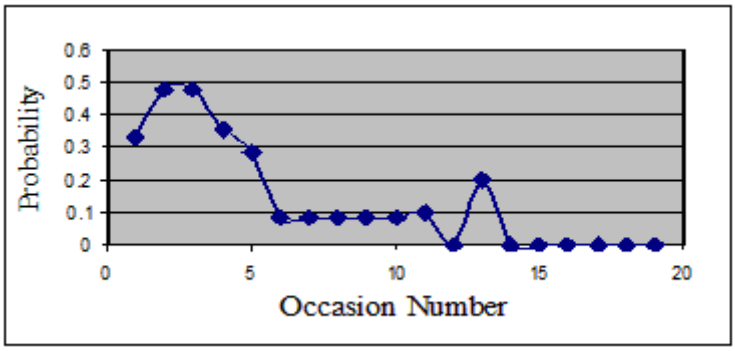


Fig. 12. Learning from dialogues.

In phase 2, five judges analyzed the interventions and indicated whether they agreed with their timing and content. At the beginning, we informed the judges that the goal of the study was to develop a model to facilitate remediation through dialogues while interacting with a tutoring system. The judges were asked to comment on the appropriateness of the timing and the content of interventions provided through the chat interface. Each judge analyzed two sessions. Due to time constraints, it was not possible to have every session investigated by two judges.

All the episodes were categorized by the rule that initiated them. Five rules were relevant in this study. Rule 1 was violated only once, and rule 2 was violated three times. The judges agreed with the timing and content of these interventions.

Rule 4 was relevant in 21 episodes and had the highest number of disagreements. Judges disagreed in 7 (33.3%) occasions. Timing was the issue in six of these instances and judges wanted to intervene earlier. The judges disagreed with the content in three situations. For instance, a judge

suggested using "*Is there a regular 1:N relationship to map in this problem*?" instead of the first prompt in Figure 10.

One of the issues to be addressed is how to effectively facilitate remediation when nothing needs to be done in a particular step (Figure 10). According to our model, the initial prompt was "*What do you need to do when you're mapping a 1:N relationship?*" which may imply that the student needs to perform an action, even if there is nothing to do. It would be better if the prompt were changed to "*Do you know which type of relationship needs to be mapped in this step*?". The new prompt discusses a domain concept so it still conforms to the dialogue structure discussed in section 3.2.

Some judges preferred earlier interventions than those suggested by the model. The model waits for 3 repeated mistakes before initiating a dialogue. However, it might be effective to intervene after two repeated mistakes, because it is easier to assess what the student is trying to achieve in this particular domain. As the result, the number of times a mistake needs to be repeated may be domain-dependent. Rule 1, which checks for a period of inactivity may also be domain-dependant; however, there was no disagreement on these.

CONCLUSIONS AND FUTURE WORK

This research focuses on developing a model to support dialogues for error-remediation in both ill- and well-defined tasks. A prototype model was developed based on the findings of a preliminary study using EER-Tutor, an ITS that teaches the ill-defined task of conceptual database design. This paper focuses on the study that used the prototype model with ERM-Tutor, an ITS developed for the ER-to-relational mapping domain. In addition to the feedback provided by the system, dialogues were facilitated through a chat interface. The interventions through the chat interface were based on the model.

Analysis of user logs indicates that students did learn the domain concepts discussed in the dialogues episodes. Human tutors who were asked to analyze the episodes mostly agreed with the interventions generated by the model.

The findings from the reported study are being used to refine the model. The next step is to incorporate the model into both EER-Tutor and ERM-Tutor. The enhanced systems will then be evaluated in authentic classroom environments. The objective of these evaluations is to investigate whether the adaptive dialogues supported by the model are more effective in facilitating deep learning than the non-adaptive dialogues, in which two students (with different domain knowledge and reasoning skills) receive the same dialogue when they make the same types of errors in their solutions.

ACKNOWLEDGEMENTS

This project was supported by the University of Canterbury doctoral scholarship to the first author. We gratefully acknowledge the support of all members of ICTG.

REFERENCES

Aleven, V., Koedinger, K.R., & Cross, K. (1999). Tutoring answer explanation fosters learning with understanding. In S. Lajoie & M. Vivet, M. (Eds.) *Artificial Intelligence in Education* (pp 199-206). Amsterdam: IOS Press.

Aleven, V., Popescu, O., & Koedinger, K.R. (2001). Towards tutorial dialogue to support self-explanation: Adding natural language understanding to a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 12, 246-255.

Aleven, V., Ogan, A., Popescu, O., Torrey, C., & Koedinger, K.R. (2004). Evaluating the effectiveness of a tutorial dialogue system for self-explanation. In J.C. Lester, R.M. Vicario & F. Paraguacu (Eds.) *Intelligent Tutoring Systems* (pp. 443-454). Berlin: Springer.

Anderson, J.R. (1993). *Rules of the Mind*. NJ: Lawrence Erlbaum Associates.

Bloom, B. (1984). The 2-sigma problem: The search of group instruction as effective as one-to-one tutoring. *Educational Researcher* ,13, 3-16.

Chen, P. (1976). The ER model - Toward a unified view of data. *ACM Transactions Database Systems*, 1(1), 9-36.

Chi, M. T. H. (2000). Self-explaining expository texts: The dual processes of generating inferences and repairing mental models. R. Glaser (Ed.) *Advances in Instructional Psychology*(pp. 161-238). NJ: Lawrence Erlbaum Associates.

Chi, M., Siler, S. A., Jeong, H., Yamauchi, T., & Hausmann, R.G. (2001). Learning from human tutoring. *Cognitive Science,* 25, 471-533.

Elmasri, R., & Navathe, S. (2007). *Fundamentals of Database Systems* (5th ed.). Boston: Addison-Wesley.

Freedman, R., Rose, C.P., Ringenberg, M.A., & VanLehn, K. (2000). ITS tools for natural language dialogue: A domain-independent parser and planner. *Intelligent Tutoring Systems* (pp. 433-442). Berlin: Springer.

Gertner, A. & VanLehn, K. (2000). ANDES: A coached problem-solving environment for physics. In G. Gauthier, C. Frasson, & K. VanLehn (Eds.) *Intelligent Tutoring Systems* (pp. 133-142). Berlin: Springer.

Goel, V., & Pirolli, P. (1988). Motivating the notion of generic design with information processing theory: The design problem space. *AI Magazine*, 10, 19-36.

Goel, V. & Pirolli, P. (1992). The Structure of Design Problem Spaces. *Cognitive Science*, 16, 395-429.

Graesser, A., Person, N., & Magliano, J. (1995). Collaborative dialog patterns in naturalistic one-on-one tutoring. *Applied Cognitive Psychology*, 9, 359-387.

Graesser, A., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R., & the Tutoring Research Group (1999). AUTOTUTOR: A simulation of a human tutor. *Journal of Cognitive Systems Research,* 1(1), 35-51.

Graesser, A., VanLehn, K., Rose, C., Jordan, P., & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue, *AI Magazine*, 22(4), 39 -51.

Jackson, G. T., Ventura, M., Chewle, P., Graesser, A. C., & the Tutoring Research Group (2004). The impact of Why/AutoTutor on learning and retention of conceptual physics. In J.C. Lester, R.M. Vicario & F. Paraguacu (Eds.) *Intelligent Tutoring Systems* (pp. 501-510). Berlin: Springer.

Jordan, P.W., Makatchev, M., Pappuswamy, U., Vanlehn, K., & Albacete, P. (2006). A natural language tutorial dialogue system for physics. In G. Sutcliffe & R. Goebel (Eds.) Proc. FLAIRS 2006, (pp.521-526) Menlo Park, California: AAAI Press.

Landauer, T.K., Foltz, P.W., & Laham, D. (1998). An Introduction to latent semantic analysis. *Discourse Process,* 25 (2-3), 259-284.

Lepper, M.R., Woolverton, M., Mumme, D.L., & Gurtner, J.L. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S.P. Lajoie and S.J. Derry (Eds.) *Computer as Cognitive Tools*. (75 -105). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Merrill, D. C., Reiser, B.J., Raney, M., & Trafton, J. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *Journal of the Learning Sciences,* 2(3), 277-305.

Milik, N., Marshall, M., & Mitrovic, A. (2006). Teaching logical database design in ERM-Tutor. In M. Ikeda & K. Ashley (Eds.) *Intelligent Tutoring Systems* (pp. 707-709). Berlin: Springer.

Millis, B., Evens, M., & Freedman, R. (2004). Implementing directed lines of reasoning in an intelligent tutoring system using the Atlas planning environment. *Proc. Information Technology* (pp. 729-733), Washington, DC: IEEE Computer Society.

Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10, 238-256.

Mitrovic, A., Suraweera, P., Martin, B., & Weerasinghe, A. (2004). DB-suite: Experiences with three intelligent, web-based database tutors. *Interactive Learning Research*, 15(4), 409-432

Mitrovic, A., & Weerasinghe, A. (2009). Revisiting the ill-definedness and the consequences of ITSs. In Dimitrova, V., Mizoguchi, R., du Boulay, B., Graesser, A (eds) *Artificial Intelligence in Education* (pp. 375-382) Amsterdam: IOS Press.

Reitman, W.R. (1964). Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In M.W. Shelly & G.L. Bryan (Eds.) *Human Judgements and Optimality (pp. 282-315)*. New York: Wiley.

Rose, C.P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive Conceptual Tutoring in Atlas-Andes. In J.D. Moore (Ed.) *Proc. Artificial Intelligence in Education* (pp. 256-266). Amsterdam: IOS Press

Rose, C.P., Kumar, R., Aleven, V., Robinson, A., and Wu. C. (2006). CycleTalk: Data driven design of support for simulation based learning, *International Journal of Artificial Intelligence in Education*, 16(2), 195-223.

Suraweera, P., & Mitrovic, A. (2002). KERMIT: a constraint-based tutor for database modelling. In S. Cerri, G. Gouarderes & F. Paraguacu (Eds.) *Intelligent Tutoring Systems* (pp. 377-387). Berlin: Springer.

Suraweera, P., & Mitrovic, A. (2004). An intelligent tutoring system for entity relationship modelling. *International Journal of Artificial Intelligent in Education,* 14(3-4), 375-417.

De Vicente, A., & Pain, H. (2002). Informing the detection of the students' motivational state: An empirical study. In S.A. Cerri, G. Gouarderes, & F. Paraguacu (Eds.) *Intelligent Tutoring Systems* (pp. 933-943). Berlin: Springer.

Weerasinghe, A., & Mitrovic, A. (2006a). Facilitating deep learning through self-explanation in an open-ended domain. *Knowledge-based and Intelligent Engineering Systems*, 10(1), 3-19.

Weerasinghe, A., & Mitrovic, A. (2006b). Individualizing self-explanation support for ill-defined tasks in constraint-based tutors. In V. Aleven, K. Ashley, C. Lynch & N. Pinkwart (Eds.) *Proceedings of the 2006 Workshop on ITS for Ill-defined Domains* (pp.56-64). Jhongli, Taiwan.

Zakharov, K., Mitrovic, A., & Ohlsson, S. (2005). Feedback micro-engineering in EER-Tutor. In C-K Looi, G. McCalla, B. Bredeweg & J. Breuker (Eds.) *Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology* (pp. 718-725). Amsterdam: IOS Press.

Revisiting Ill-Definedness and the Consequences for ITSs

Antonija MITROVIC and Amali WEERASINGHE
Intelligent Computer Tutoring Group
University of Canterbury, Private Bag 4800, Christchurch, New Zealand
tanja.mitrovic@canterbury.ac.nz, amali.weerasinghe@pg.canterbury.ac.nz

Abstract: ITSs for ill-defined domains have attracted a lot of attention recently, which is well-deserved, as such ITSs are hard to develop. The first step towards such ITSs is reaching a wide agreement about the terminology used in the area. In this paper, we discuss the two important dimensions of ill-definedness: the domain and the instructional task. By the domain we assume declarative domain knowledge, or the domain theory, while the instructional task is the task the student is learning, in terms of problem-solving skills. It is possible to have a well-defined domain and still have ill-defined instructional tasks in the same domain. We look deeper at the features of ill-defined tasks, which all contribute to their ill/well defined nature. The paper discusses model-tracing and constraint-based modeling, in terms of their suitability for ill-defined tasks and domains. We show that constraint-based modeling can be used in both well- and ill-defined domains, and illustrate our conclusion using several instructional tasks.

Introduction

Recently there has been a lot of attention on supporting learning in ill-defined domains (aka ill-structured domains), as evidenced by the three workshops held in 2006-8 [1-3]. This attention is welcome, as ITSs for such domains are rare, and usually much more demanding than the typical ITSs for well-defined domains. However, there has been little agreement about the terminology used and the underlying definitions between researchers working in this area, even among those who presented their work at the workshops. Most researchers equate ill-defined domains with ill-defined tasks [1-3]. In this paper, we argue that it is important to make the distinction between instructional domains and tasks. We start by discussing instructional domains and tasks as two important dimensions in the light of ITSs. Section 2 presents a deeper discussion of instructional tasks, focusing on various factors which influence the nature of the task. We then turn to student modeling approaches which are appropriate for various instructional situations, and then show how ITSs can deal with ill-definedness.

1. A deeper look at ill-definedness: the two dimensions

An instructional domain is an area of study, such as mathematics or philosophy. In order to learn a particular instructional domain, the student needs to learn the relevant declarative knowledge (i.e. the domain theory), and in many domains also needs to

acquire problem-solving skills. ITSs are almost exclusively problem-solving environments, based on the assumption that students have learnt the declarative knowledge from direct instruction (lectures, books and/or peers) and only need to practice their problem-solving skills [4, 5]. Most ITSs provide lots of problem-solving opportunities and only occasionally give direct instruction, in the form of examples or definitions of the concepts used as in [6]. There are also ITSs that provide instructional material in addition to problem-solving support, such as ELM-ART [7], but in this paper we will focus on problem-solving as the main instructional activity.

It is important to make a clear distinction between those two types of learning (acquiring declarative knowledge versus problem-solving skills) for the discussion of ill-definedness. We found a lot of confusion in published papers when discussing ill-definedness. Most researchers equate ill-definedness with the underlying domain theory, and provide examples of ill-defined domains, such as essay writing. Commonly used examples of well-defined domains are mathematics and physics. However, there seems to be no differentiation between the characteristics of domains versus tasks.

We propose that two orthogonal dimensions need to be considered when discussing ill-definedness: the domain, and the task. Starting from our first dimension, domains vary in terms of their underlying domain theories. There are many domains covered by ITSs that are completely well-defined, such as many areas of mathematics, physics and chemistry. Instructional tasks that they teach are also well-defined: for example, adding fractions, solving equations for unknowns, or balancing chemical equations. The student is taught the theory, as well as the procedure (i.e. the algorithm) to use to solve problems. Such domains are in the WDWT quadrant in Figure 1.

However, if the domain is well-defined, that does not necessarily mean that instructional tasks in that domain will also be well-defined. As an illustration, let us focus on the domain of database design [8]. Conceptual database design is a task of converting the database requirements into a high-level description of the database, most often expressed in terms of the Entity-Relationship (ER) data model [8]. On the other hand, logical database design is a process of converting the ER diagram into a relational schema, thus requiring an understanding of the relational data model. Both the ER and relational data models are well-defined: they consist of a small number of components with well-defined syntax and semantics. Although the ER model itself is well-defined, the task of developing an ER schema for a particular database (i.e. conceptual database design) itself is ill-defined: the initial state (i.e. the set of requirements) is usually underspecified and ambiguous, there is no algorithm to use to come up with the solution, and finally the goal state is also underspecified, as there is no simple way of evaluating the solution for correctness. Therefore, conceptual database design belongs to the WDIT quadrant in Figure 1. Logical database design, however, is well-defined, as there is a simple deterministic algorithm which guarantees good solutions (shown in the WDWT quadrant in Figure 1). Other examples for the WDIT quadrant include programming and writing SQL queries: although the relevant languages are well-defined, the task of converting the problem statement into a program is ill-defined.

Many domains are ill-defined, such as essay writing. In that case, the declarative knowledge is incomplete: it specifies how to structure the essay, how to present arguments, and also defines writing styles. The domain theory in this case is ill-defined, as is the task itself (writing the essay), as illustrated in the IDIT quadrant in Figure 1. The two dimensions are continuous; there is a spectrum arranging domains from ill- to well-defined ones, as well as another spectrum for instructional tasks. There

are some dependencies between them, as ill-defined domains usually involve ill-defined tasks, but the contrary is not necessarily so. Note that there are no examples for the IDWT quadrant: here the domain theory is ill-defined, but the task is well-defined. We believe this combination is not possible, and do not consider it further.

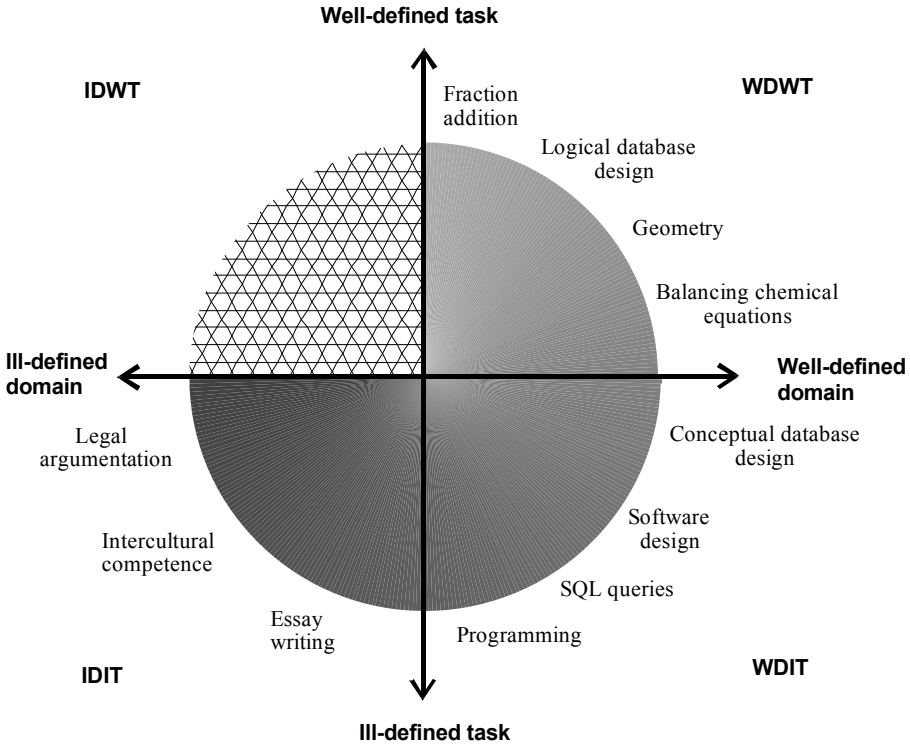


Figure 1. The space of instructional domains and tasks

2. Classifying instructional tasks: important characteristics

When discussing the definedness of decision-making tasks, there are four important factors to consider [9, 10]: start state, goal state, and the transformations (i.e. the problem-solving procedure), as well as the decision maker's familiarity with each of the factors. Decision making is similar to problem solving, and for that reason we adopt those factors. In addition, we add another one: the existence of a correct solution.

The initial state is presented to the student in the form of a problem statement. Instructional tasks taught to younger students most often have well-specified problems statements – e.g. simple arithmetic tasks, equation solving and other tasks in science. Problem statements for more challenging tasks can be less specified: in a typical university-level mechanics problem, the text of the problem does not specify all the forces acting on a given body. In conceptual database design or software design, the student is given a set of requirements, which is often incomplete and/or ambiguous. To deal with such problems, the student needs to use not only declarative domain knowledge they learnt previously, but also his/her world knowledge in order to

eliminate ambiguities and (when necessary) add missing information. Therefore, in order to deal will ill-defined problem statements, the student has to process the given information in order to complete the specification (and therefore turn the problem statement into a well-defined one).

Goal states can also be well- or ill-defined. In easy tasks, the student is clear about the form of the final solution. For example, the student had learnt that there were two solutions for a quadratic equation before attempting to solve any equations. When adding two fractions, the student knows that the solution should be (in the general case) another fraction. Additionally, the student can easily check whether the solution is correct or not. However, in design tasks, there is little information about the goal state. The goal state in such tasks is defined in a very abstract way; for example, in database design the goal state is defined as an ER diagram that is syntactically correct and matches the given requirements. Therefore, there is no simple test to use to check for correctness; the student can only apply the declarative knowledge he/she possesses in order to evaluate the solution produced. Another important issue is whether there is a stopping criterion – how can the student tell whether he/she is finished solving the problem? A well-defined goal possesses a stopping criterion which is easy to apply, while the ill-defined ones do not, and the student is again left to apply the constraints from the declarative knowledge in order to evaluate the solution.

Transformations or the problem solving procedure is another important factor. Some instructional tasks have a well-defined algorithm to apply to the initial state to derive the goal state. We have previously mentioned several tasks from mathematics, physics or chemistry with well-defined algorithms. In such situations, the student task is (relatively) easy: the student needs to memorize the algorithm and apply it correctly. Other similar examples involve some engineering tasks involving calculations. However, there is a very important subclass of tasks that deal with design. Design is in general ill-defined, as there are no algorithms to use to transform the initial state into the goal state. In addition, the start state is underspecified, and the goal state defined in terms of highly abstract features. Design tasks typically involve huge domain expertise, and large, highly structured solutions [11]. Typical examples of design tasks include architecture, software design, mechanical engineering and music composition. In such tasks there may be heuristic rules that can guide the student, but in general the student needs to apply the constraints from the domain theory. The other possible mechanism is analogy: the student can compare the current problem to those previously solved and perform analogical reasoning to deal with the complexity.

Finally, some researchers believe that ill-defined tasks are those that have no correct solution, but rather a family of solutions which can all be deemed correct. This is true of the extreme cases, such as essay writing: there might be any number of very good essays on a specified topic. In design tasks, there are often several (or even many) solutions that are all equally good. However, in a teaching situation, the teacher often has a good pedagogical reason for preferring one particular solution over the others. For example, in SQL there are often several correct queries for the same problem, differing from each other in the constructs used (please note that there is a lot of redundancy in SQL and, therefore, multiple ways of satisfying the same requirements). Even in such a task, the teacher may prefer one of those solutions among others; for example, the teacher may want to illustrate the use of a particular predicate. Therefore, it is still possible to nominate one “ideal” solution without compromising the quality of the whole ITS, as long as the ITS is capable of identifying other alternative solutions students may come up with as correct.

Table 1 presents the two dimensions and the factors of instructional tasks, and presents a few examples, categorized with respect to the factors discussed.

Table 1. Some examples of instructional tasks and their domains

Instructional task	Domain	Problem specification (initial state)	Goal specification (goal state)	Problem-solving procedure	Correct solution
Fraction addition	Well-defined	Well-defined	Well-defined	Well-defined	Only one
Balancing chemical equations	Well-defined	Well-defined	Well-defined	Well-defined	Only one
SQL queries	Well-defined	Ambiguous/ incomplete	Abstract	None	Multiple
Software design	Well-defined	Ambiguous/ incomplete	Abstract	None	Multiple
Essay writing	Ill-defined	Abstract	Abstract	None	Multiple
Legal argumentation	Ill-defined	Abstract	Abstract	None	Multiple
Intercultural competence	Ill-defined	Abstract	Abstract	None	Multiple

3. Student modeling approaches and ill-definedness

Another important issue is how the ITS models the student. Model tracing [4] is the most widely used student modeling approach currently. Since it tracks student's progress by generating solutions step-by-step, this approach is suited to well-defined tasks. Developing model-tracing tutors for ill-structured tasks is much harder, as it is difficult to come up with problem solvers for such tasks [1-3, 12]. However, constraint-based tutors do not suffer from such difficulties. Within the Intelligent Computer Tutoring Group (ICTG), we have developed constraint-based tutors for many tasks, both well- and ill-defined. Examples of our tutors teaching well-defined tasks range from fraction addition and balancing chemical equations to data normalization in relational databases [5]. We have also been very successful in developing ITSs for ill-defined, design tasks. SQL-Tutor [13] is our first ITS that teaches students how to formulate queries in SQL. The typical problem statement specifies the type of data the query is to return, and is often ambiguous. The student needs to understand the database structure and the data that is stored in the database, as well as the relational data model and the SQL constructs. The transformation for the problem statement into an SQL query is underspecified; the students are taught search strategies on examples, and need to be able to use analogies with previously solved queries to solve new ones, as well as to use the SQL syntax. EER-Tutor [5] (the early version of which was known as KERMIT [14]) teaches conceptual database design, the task previously discussed in section 1. COLLECT-UML is another ITS that teaches a design task, this time object-oriented software design using UML class diagrams [15]. In addition to teaching UML class diagrams, COLLECT-UML also teaches collaboration skills.

Our tutors are based on Constraint-Based Modeling (CBM), a domain and student modeling approach that represents domain knowledge as a set of constraints on correct solutions [16]. Constraints capture both syntax and semantics of a domain, and are very computationally efficient. In our previous work, we have argued that CBM is ideally suited to teaching ill-defined tasks [12]. Constraint-based tutors do not require the problem solver, as they can evaluate the student's solution by comparing it to a solution pre-specified by a teacher. As discussed previously, it is not unrealistic to expect the

teacher to specify one preferred solution for design tasks, as such a solution is motivated pedagogically. Therefore our tutors check the semantics of the student's solution by comparing it to the pre-specified ideal solution. At the same time, they are capable of identifying alternative solutions as correct, as constraints check for equivalent ways of solving the problem.

Goel and Pirolli [17] argue that design problems by their very nature are not amenable to rule-based solutions (as in model tracing). On the other hand, constraints are extremely suitable for representing design solutions: they are declarative, non-directional, and can describe partial or incomplete solutions. A constraint set specifies all conditions that have to be simultaneously satisfied without restricting how they are satisfied. Therefore, the ITS performs the same process the student needs to perform in order to evaluate his/her solution – apply domain constraints to it.

4. How can ITSs support learning in IDIT?

We argue that CBM is suitable for use in both well- and ill-defined domains/tasks. We have discussed examples of constraint-based tutoring systems that work in the two quadrants in Figure 1 corresponding to well-defined domains (WDWT and WDIT), with well- or ill-defined tasks. Can CBM also be used in the IDIT quadrant?

To answer this question, let us discuss an example instructional task that belongs to this quadrant. Walker et al. [18] describe one such situation: teaching intercultural competence, a task in which the student needs to explain observed cultural behaviour. This is an ill-structured task, as the start/end goals are ill-defined, there is no algorithm to use to solve the problem, and additionally there is no stopping criterion. The domain theory is also ill-defined; certain norms and rules are known about a culture involved, but there are many exceptions to them and also personal differences make the whole process very hard. Model tracing cannot be used in this case, as it is hard (or maybe even impossible) to come up with the cognitive model of the task. In [18], whatever is known about the domain is captured in terms of five dimensions: the student's discussion needs to be on topic, must be based on provided facts, needs to contain multiple perspectives and a good argument for the claims and observations. The fifth dimension allows the student to also provide novel facts to support their argument, which have not been provided in the case. Walker et al. assess student's discussion by comparing it to the model of a good discussion – in essence this model is an "ideal" solution. Since the domain and the task are both ill-defined, it is impossible to specify the ideal solution completely; in such situations, the ideal solution consist of *mandatory* elements, and other elements are optional, and depend on the personal preference.

Walker et al. [18] effectively do what constraint-based tutors do: they created a mechanism for comparing the student's solution to the ideal solution provided by the teacher. They also provide feedback to the student, saying what is good in his/her discussion, and suggesting how to improve. CBM can be applied in this case: the constraint set would consist of the syntax restrictions that must be satisfied, and the semantic constraints would check that the solution is consistent with the given problem. This implies another possible design for constraint-based tutors: the one in which the ideal solution would be replaced with a formal specification of problem requirements.

Another illustration is the domain of legal reasoning, as done in LARGO [19]. In this system, the student needs to develop a diagram reflecting the legal case. This diagram is then compared to the expert's solution, and feedback is provided about

wrong or missing elements. Architectural design also belongs to this group. If the task is to design a house with three bedrooms for a given piece of land which needs to be eco-friendly and energy efficient, there can be a set of designs which satisfy the minimal requirements. Constraints that need to be satisfied involve the problem specification and the norms for energy consumption and ecological consequences – but the designs will differ in terms of aesthetics and personal preferences of the designer. Again, the constraint set will capture the minimal requirements, and still allow for a variety of solutions. Therefore, in ill-defined domains the student has the freedom to include solution components to make the solution aesthetically pleasing or more to their preferences, and the ITS will still accept it as a good solution for the problem. It is also possible to have weights attached to constraints, with highest weights being assigned to mandatory constraints, and lower weights assigned to constraints that need not necessarily be satisfied as they correspond to optional elements.

In IDIT, more attention needs to be devoted to the feedback provided to the student. In well-defined domains, feedback generation is straightforward: the student violates some constraints, and feedback on violated domain principles is provided. In model-tracing tutors, buggy production rules provide feedback on errors, and hints can be generated on the next step the student is to take. However, in ill-defined domains, the declarative knowledge is incomplete: the constraint set consists of a set of mandatory principles and some heuristics. Therefore, the feedback mechanism needs to be sophisticated, so that feedback does not confuse the student. Walker et al. [18] provide suggestions to the student, based on prioritized dimensions they identified, which is in essence similar to the mechanism we use to select a tutorial dialogue in KERMIT [20]. If the solution is a partial one, feedback becomes even more crucial, as the ITS should discuss only the issues the student has worked on so far.

Ill-defined domains and tasks are very complex, and therefore, ITSs need to scaffold learning, by providing as much information as possible without making it trivial. The common ITS techniques can also be used in ill-defined domains (e.g. visualizing goal structure and reducing the working memory load [13], providing declarative knowledge in the form of dictionaries or on-demand help [6, 7], etc). Furthermore, the ITS can simplify the process by performing one part of the task for the student automatically [21] or by restricting the actions students can take [4]. Furthermore, solution evaluation can be replaced with presenting consequences of student actions [22] or supporting a related, but simpler task, e.g. peer review [23].

5. Conclusions

This paper proposes two important dimensions for discussing ill-definedness, which can be used to order the domains/tasks along a continuum, from well- to ill-defined. As designers of ITSs, we cannot do much about the domain definedness – it is either well- or ill-defined (in other words, the domain theory either exists or does not exist). However, the features of instructional tasks are very important. We discussed the definedness of the initial state (i.e. the problem statement) and final, goal state (i.e. the form of the solution), as well as the problem-solving procedure and the number of alternative solutions. Instructional tasks vary on all of those four factors.

In previous work, we have shown that CBM can be used effectively to support learning in well-defined domains, with either well- or ill-defined tasks. CBM can deal with ill-defined tasks, as each constraint tests a particular aspect of the solution, and

therefore supports modularity. Incremental development is supported by being able to request feedback at any time. Therefore, CBM can evaluate partial solutions: if a particular part of the solution is incomplete, the student will be informed about that.

We also show that CBM can support ill-defined domains, by discussing current research on such domains, and identifying similarities with constraint-based tutors. Constraints can capture whatever is known about the ill-defined domain and the problem specification, thus begin able to evaluate the mandatory parts of the solution. Such a tutor can provide feedback to student, while still allowing for multiple solutions differing in non-essential elements, such as aesthetical and personal preferences.

References

[1] V. Aleven, K. Ashley, C. Lynch, N. Pinkwart (eds) ITS 2006 Workshop on Intelligent Tutoring Systems for Ill-Defined Domains, 2006.

[2] V. Aleven, K. Ashley, C. Lynch, N. Pinkwart (eds) AIED 2007 Workshop on AIED applications in ill-defined domains, 2007.

[3] V. Aleven, K. Ashley, C. Lynch, N. Pinkwart (eds) ITS 2008 Workshop on Intelligent Tutoring Systems for Ill-Defined Domains: assessment and feedback in ill-defined domains, 2008.

[4] K. Koedinger, J. Anderson, W. Hadley, and M. Mark, Intelligent tutoring goes to school in the big city, *Int. Journal of Artificial Intelligence in Education* **8** (1997), 30-43.

[5] A. Mitrovic, B. Martin, P. Suraweera, Intelligent tutors for all: Constraint-based modeling methodology, systems and authoring. *IEEE Intelligent Systems*, **22** (2007), 38-45.

[6] V. Aleven, K. Koedinger, K. Cross, Tutoring answer explanation fosters learning with understanding. In S. Lajoie, M. Vivet (eds) Artificial Intelligence in Education, pp. 199-206, 1999.

[7] G. Weber, P. Brusilovsky, ELM-ART: an adaptive versatile system for Web-based instruction. *Int. J. Artificial Intelligence in Education* **12** (2001), 351-384.

[8] R. Elmasri, S. Navathe, *Fundamentals of Database Systems*. Addison Wesley, 2004.

[9] W.R. Reitman, Heuristic decision procedures, open constraints, and the structure of ill-defined problems. In: M.W. Shelly, G.L. Bryan (eds) *Human judgements and optimality*, pp. 282-315, 1964.

[10] R.N. Taylor, Nature of problem ill-structuredness: implications for problem formulation and solution. *Decision Sciences* **5** (1974), 632-643.

[11] V. Goel, P. Pirolli, The structure of design problem spaces. *Cognitive Science* **16** (1993), 395-429.

[12] S. Ohlsson, A. Mitrovic, Fidelity and Efficiency of Knowledge representations for intelligent tutoring systems. *Technology, Instruction, Cognition and Learning* **5** (2007), 101-132.

[13] A. Mitrovic, S. Ohlsson, Evaluation of a Constraint-Based Tutor for a Database Language. *Int. J. Artificial Intelligence in Education* **10** (1999) 238-256.

[14] P. Suraweera, A. Mitrovic, An Intelligent Tutoring System for Entity Relationship Modelling. *Int. J. Artificial Intelligence in Education* **14** (2004), 375-417.

[15] N. Baghaei, A. Mitrovic, W. Irvin, Problem-Solving Support in a Constraint-based Intelligent Tutoring System for UML. *Technology, Instruction, Cognition and Learning*, **4** (2006), 113-137.

[16] S. Ohlsson, Constraint-based Student Modelling. *Proc. of Student Modelling: the Key to Individualized Knowledge-based Instruction*, Springer-Verlag, Berlin, (1994) pp. 167-189.

[17] V. Goel, P. Pirolli, Motivating the notion of generic design with information processing theory: the design problem space. *AI Magazine* **10** (1988), 19-36.

[18] E. Walker, A. Ogan, V. Aleven, C. Jones, Two approaches for providing adaptive support for discussion in an ill-defined domain. In [4], pp. 1-12.

[19] N. Pinkwart, V. Aleven, K. Ashley, C. Lynch, Evaluating Legal Argument Instruction with Graphical Representations Using LARGO. Proceedings of AIED 2007, pp. 101–108.

[20] A. Weerasinghe, A. Mitrovic, Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. *Knowledge-based and Intelligent Engineering Systems* **10** (2006), 3-19.

[21] M. Easterday, V. Aleven, R. Scheines, The logic of Babel: causal reasoning from conflicting sources. In [3], pp. 31-40, 2007

[22] R. Hodhod, D. Kudenko, Interactive narative and intelligent tutoring for ethics domain. In [4], pp. 13-, 21, 2008.

[23] I. Goldin, K. Ashley, R. Pinkus, Teaching case analysis through framing: prospects for an ITS in an ill-defined domain. In [2], pp. 83-91, 2008.

S. L. Wong et al. (Eds.) (2010). Proceedings of the 18th International Conference on Computers in Education. Putrajaya, Malaysia: Asia-Pacific Society for Computers in Education.

Evaluating the Effectiveness of Adaptive Tutorial Dialogues in EER-Tutor

Amali WEERASINGHE, Antonija MITROVIC, Martin VAN ZIJL, Brent MARTIN
Intelligent Computer Tutoring Group, University of Canterbury, New Zealand
amali.weerasinghe@pg.canterbury.ac.nz

Abstract: Researchers have long been interested in tutorial dialogues as they are considered to be one of the critical factors contributing to the effectiveness of human one-on-one tutoring. We discuss an evaluation study that investigates the effectiveness of adaptive tutorial dialogues in database design. EER-Tutor, a database design tutor was enhanced to facilitate adaptive tutorial dialogues. The control group participants received non-adaptive dialogues regardless of their knowledge level and explanation skills. The experimental group participants received adaptive dialogues that were customised based on their student models. The performance on pre- and post-tests indicated that the experimental group participants learned significantly more than their peers. The subjective responses indicated no difference in their impression towards the quality of the dialogues and the understandability of the questions. However there was clear evidence that the control group did not like having to go through the entire dialogue before resuming problem-solving.

Keywords: tutorial dialogues, constraint-based tutors, adaptive dialogues

Introduction

One-on-one human tutoring is wildly considered to be the most effective form of instruction [2]. Students' learning gains have been increased by two standard deviations when tutored by human tutors compared to traditional classroom instruction. This has inspired researchers to explore how the effectiveness of human tutoring strategies can be incorporated into intelligent tutoring systems. One of the critical factors contributing to the effectiveness of human tutoring is the conversational aspect of the instruction. Dialogues provide opportunities for students to reflect on their existing knowledge and to construct new knowledge. Some of the dialogue-based tutoring systems that have been developed are Why2-Atlas [3], Auto Tutor [3], CIRCSIM-Tutor [4], Geometry Explanation Tutor [1] and KERMIT-SE [9]. Why2-Atlas and AutoTutor use dialogues as the main activity to help students learn the domain knowledge. The other systems provide problem-solving environments as the main activity and use tutorial dialogues as a way of remediating errors in the student solutions. For example, CIRCSIM-Tutor is a natural language (NL) tutor that helps students learn cardiovascular physiology related to regulation of blood pressure. The Geometry Explanation Tutor requires students to justify the problem-solving steps in their own words. KERMIT-SE, a database design tutor, engages students in dialogues when their solutions are erroneous. All these instructional tasks except database design are well-defined: problem solving is well-structured, and therefore explanations expected from learners can be clearly defined. In contrast, database design is an ill-defined task: the final result is defined only in abstract terms, and there is no algorithm to find it.

Our long-term goal is to develop a general model for supporting dialogues across domains. Since we previously implemented dialogues for EER-Tutor [5], the initial work on this project started with the same system. Based on the findings of two Wizard-of-Oz studies [7, 8], we developed a model to support dialogues. Our model consists of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all the error types in a domain. At the lowest level an error type is associated with one or more violated constraints, which form leaves of the hierarchy. The error types

are then grouped into higher-level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue learners are able to provide answers by selecting the correct option from a list provided by the tutor. For a detailed discussion of the model see [7].

The next section presents the details of the evaluation study. Section 2 presents the results followed by conclusions.

1. The Study

We conducted a study with the EER-Tutor in March 2010 at the University of Canterbury, which involved volunteers from an introductory database course. The objective of the study was to investigate whether adaptive dialogues are more effective in improving learning than non-adaptive dialogues. The participants were randomly assigned to two groups (experimental and control). The experimental group received adaptive support based on our model. The control group was given non-adaptive support in which two different students with different knowledge levels received the same dialogue. Differences between the two groups were: (i) Dialogue selection (ii) Dialogue prompts and (iii) Additional support.

Dialogue selection: The dialogue selection for the control group was based on a depth-first traversal of the error hierarchy. The first violated constraint that was found in the traversal was selected for discussion. As the errors in the hierarchy were ordered from simpler to more complicated errors, the depth-first search results in the simplest error to initiate a dialogue. For instance, Figure 1 presents the dialogue that a control group participant receives for the submitted solution. It contains multiple errors: (i) ROOM should be represented as a weak entity instead of a regular entity (ii) Attributes are missing from the entities HOTEL, EMPLOYEE and ROOM (iii) Cardinality between HOTEL and WORKS_FOR is incorrect etc. The error selected for discussion was that ROOM was modelled as a regular entity. Now consider an experimental group participant with an identical student model to the previous student submitting the same solution. Figure 2 represents the dialogue to be received. This dialogue focuses on the incorrect cardinality between EMPLOYEE and HOTEL. This is because cardinality was identified as the most difficult concept based on his/her student model. (i.e. the error this student will most likely to make in the next attempt).

Dialogue prompts: The control group saw the entire dialogue regardless of the number of times they have seen the dialogue previously or their responses to the dialogue prompts. As a result, the same solution submitted by two different students with different knowledge levels in the control group received identical dialogues. For instance, the prompt received by the control group participant discusses the domain concept related to the error selected for discussion (EERTutor1 in Figure 3(a)). We call this type of prompt a problem-independent prompt as it focuses on the relevant domain concept [7]. The entire dialogue is given in Figure 3(a). In contrast, the prompt received by the experimental group participant discusses the selected error in the context of the current problem (EERTutor3 in Figure 3(b)). This type of prompt is called problem-dependent prompt [7]. This error was chosen for discussion because his/her student model identifies that cardinality is the most difficult concept at this moment. He/She receives a problem-dependent prompt (instead of a

problem-independent prompt) because this is the first time this mistake is made during the current session. If he makes this type of error repeatedly, he will be given the problem-independent prompt (EER-Tutor1 in Figure 3(b)).

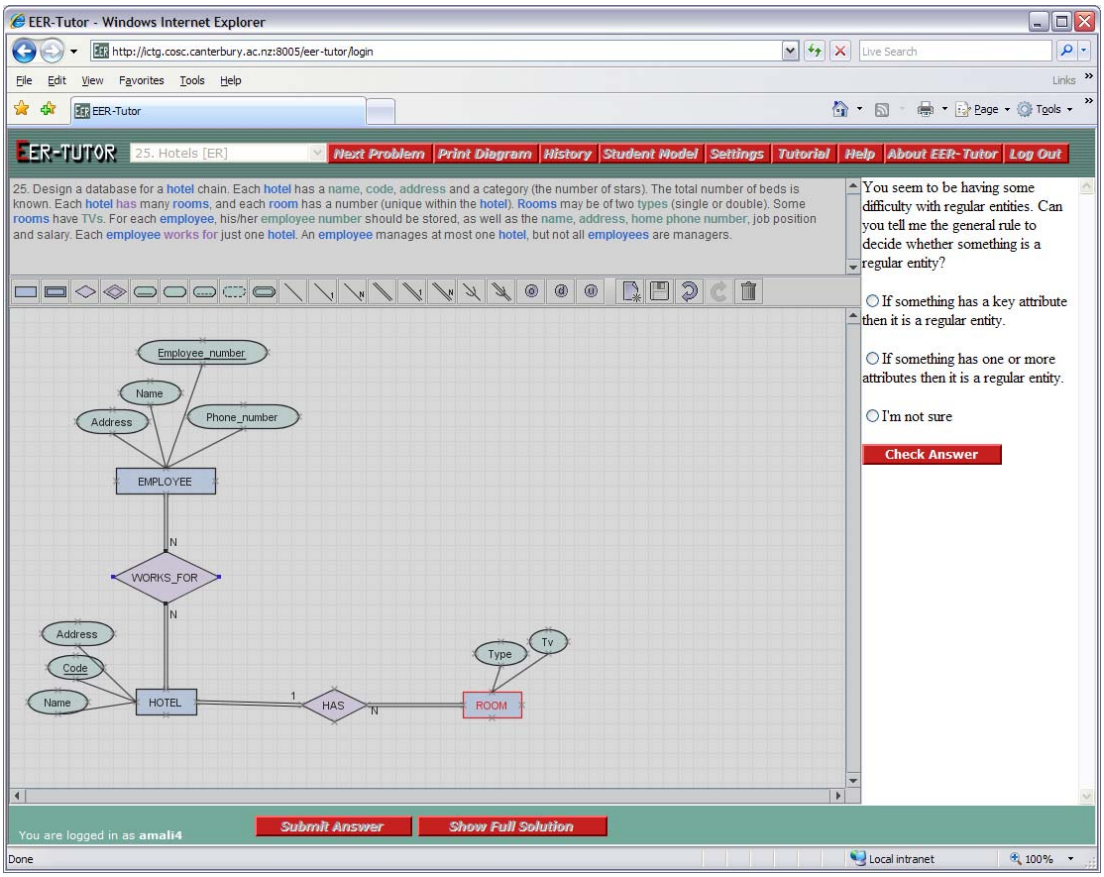


Fig. 1. The dialogue received by a control group participant

Additional support: When an experimental group participant abandons a problem (i.e. changes a problem without submitting at least once) or has been inactive for a period of time, they were asked whether they needed help. If they requested help then their solution was evaluated and an error was selected for discussion based on their student model. The control group did not receive this support.

The study consisted of four stages: (i) pre-test (ii) interactions with EER-Tutor (iii) post-test (iv) questionnaire.

Pre- and post-tests: Pre-tests were used to determine the participants' knowledge before interacting with the system and also to determine whether the knowledge between the experimental and control was significantly different. Both pre- and post-tests had 6 questions each. The questions in the pre- and post-tests were of similar difficulty. We wanted to evaluate whether students' problem-solving abilities as well as explanation skills improved after interacting with the system. One question asked the participants to provide the database schema for the given requirements. This is a typical question that can be found in examinations, text books etc. Three other questions were aimed to understand the effect

the system had on students' explanation skills. The remaining two questions asked about declarative knowledge.

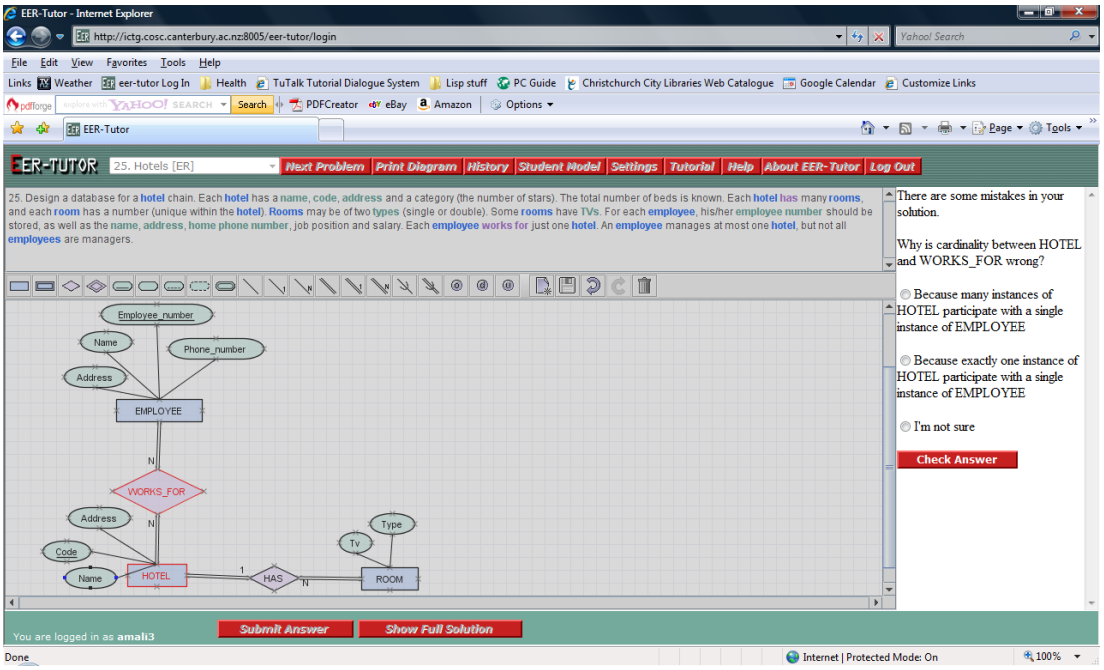


Fig 2. The dialogue received by an experimental group participant

Questionnaire: There were eight questions aimed at understating participants' view of different aspects of dialogues. Five questions had Likert scales (ranging from 1 to 5) discussing the quality, the length and the prompts in the dialogues. Participants were also given an opportunity to explain how the dialogues helped them in their learning. Some questions focused on suggestions to improve the dialogues and EER-Tutor in general.

Procedure: The participants used EER-Tutor for the first time in their regular lab sessions during the third week of the course, by which time they had been introduced to ER modeling.
They were randomly divided into the control and experimental groups. The duration of the study was limited to a single 2-hour session. At the beginning of the session students were given about 10 minutes to complete the pre-test. Then they started solving ER modelling problems using EER-Tutor. Towards the end of the session, they were given 10 minutes to complete the post-test and 5 minutes to answer the questionnaire.

2. Results and Discussion

Out of 104 students enrolled in the course, 77 participated in the study. Some students did not complete the post-test. Table 1 reports some statistics about the 65 participants who completed both pre-and post-tests. There were 31 participants in the experimental group and 34 in the control group. There was no significant difference on the pre-test performances of the two groups. The post-test performance of the experimental group was significantly better compared to their peers who received non-adaptive dialogues Both the learning gain[1]

[1] Learning gain = post-test score – pre-test score

and the normalised learning gain[2] of the group who received adaptive dialogues was also significantly higher than the gains of the control group.

EERTutor1: You seem to be having some difficulty with regular entities. Can you tell me the general rule to decide whether something is a regular entity? ***Student1***: *I don't know* **EERTutor2:** Let's try a simple thing. What is used to uniquely identify instances of a regular entity ***Student2***: key a*ttribute* **EERTutor3 :** Great ! Now tell me why modelling *ROOM* as a regular entity is incorrect? ***Student3***: *I'm not sure* **EERTutor4**: Let's see. Check carefully whether you can find a key attribute for *ROOM*. {When Continue button is pressed} Now please select the key attribute for *ROOM* ***Student4***: *No key attribute for ROOM* **EERTutor5**: Great! Now tell me the best way to model *ROOM*? ***Student5***: *I need more help* **EERTutor6:** Ok. As *ROOM* does not have a key attribute it cannot be a regular entity. So it should be a weak entity. **EERTutor7**: Great ! try to answer this question so that you'll remember what you learnt. …….. uniquely identifies instances of a …… ***Student6***: key *attribute, regular entity* **EERTutor8**: Excellent, go ahead and do the changes. Submit the solution when you're done.	**EERTutor1**: Looks like you have trouble with cardinality. Let's focus on cardinality now. What do you represent by cardinality on the E1 side between entities E1 and E2 in a binary relationship? ***Student1***: *I don't know* **EERTutor2:** Here's a simple question to try. How do we model connections between entities? ***Student2***: *Using relationships* **EERTutor3 :** Great ! Why is the cardinality between *HOTEL and WORKS_FOR* is wrong? ***Student3***: *I'm not sure* **EERTutor4**: Let's see. Try to figure out how many instances of *HOTEL* participates with a single instance of *EMPLOYEE*? {When Continue button is pressed}Now tell me how many instances of *HOTEL* participates with a single instance of *EMPLOYEE*? ***Student4***: Exactly one **EERTutor5:** Great! Guess you know how to make change now. Before that, try to answer this question. What is the correct question to ask when deciding the cardinality on the E1 side between entities E1 and E2 in a binary relationship? ***Student5***: How many instances of E1 participates with a single instance of E2? **EERTutor6**: Excellent job, go ahead and do the changes. Submit the solution when you're done.

Fig. 3 A hypothetical dialogue for EER-Tutor received by a (a) control group participant (b) experimental group participant

As the study was limited to a single lab session, the two groups spent a similar time interacting with the system. There were also no significant differences between the number of attempted and solved problems. The total number of dialogues, the total number of single-level dialogues (some dialogues are limited to a single feedback message as they discuss simple errors) and the total number of multi-level dialogues were also similar for the two groups.

The control group answered a significantly higher number of questions than their peers. This was expected, as the control group had to go through the entire dialogue before resuming problem-solving. However, percentage number of correct answers was similar for both groups. There were no significant differences on the total number of questions answered incorrectly or the questions with a *More Help* request (i.e one of the options available was *I don't know* or *I need more help* which resulted in presenting the relevant information to the

[2] Normalised learning gain =learning gain/(1-pre-test score)

student). Also there was no significant difference on the percentage of questions that requested more help. However, it is interesting to note that the experimental group has provided a significantly higher percentage of incorrect answers. Further analysis is required to understand the cause for this.

Table 1. Some statistics from the study (sd given in parentheses)

	Control (34)	Experimental (31)	p
Pre-test (%)	54.5 (18.1)	51.3 (16.1)	ns
Post-test mean (%)	61.2 (14.9)	69.9 (11.5)	0.005
Gain	6.8(15.6)	18.6 (16.8)	0.002
Normalised gain	0.002 (0.7)	0.3 (0.4)	0.01
No. of constraints learnt	1.2 (1.5)	2.3 (2.3)	0.02
Interaction time (min)	62.8 (22.1)	62.9 (24.1)	ns
Attempted Problems	8.6(4.8)	10.6(4.8)	ns
Solved problems	9.0(4.8)	7.9 (4.7)	ns
Total Dialogues received	12.1 (7.3)	14.0 (8.3)	ns
Single-level dialogues seen	2.1(3.0)	1.9 (2.7)	ns
Multi-level dialogues seen	10 (6.8)	12.1(7.2)	ns
Total number of questions answered	34.4 (25)	23.6 (14.6)	0.01
Total number of questions answered correctly	23.3 (17.9)	14 (10.4)	0.006
% number of questions answered correctly	61.4(23.1)	59(16.9)	ns
Total number of questions answered incorrectly	9.1 (8.3)	7.3 (4.3)	ns
% number of questions answered incorrectly	23.7(12.9)	31.8(15)	0.01
Total number of questions with a *More Help* request	2.1 (3.5)	2.4 (3.5)	ns
% number of questions answered with a *More Help* request	6.1(6.9)	9.22(11.4)	ns

Effect size[3] is a standard way to compare the results of one pedagogical experiment to another. It indicates how much more the experimental group has learnt compared to the control group? The effect size (Cohen's d) for learning gains of the two groups is 0.69 (the effect size based on the normalized gain is 0.51). This is comparable to the study with SQL-Tutor conducted in a similar setting in a single 2-hour session [6]. An effect size of 0.66 was reported for that study for the students who used SQL-Tutor compared with those who did not use the tutor. The effect size obtained here is therefore remarkable because the only difference between the two groups was the adaptivity of the dialogues.

2.1 Learning Curves

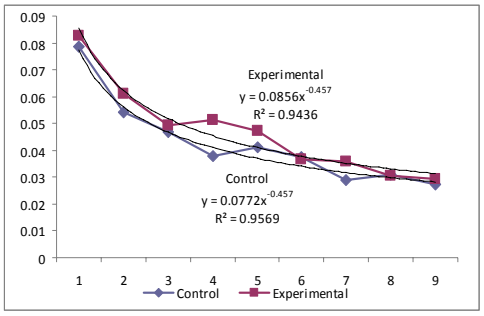


Fig 4: Probability of violating a constraint

In order to investigate how the students in both groups learnt the database design concepts in terms of constraints we analyzed how frequently constraints were violated. Figure 4 illustrates the learning curves for both groups. The probabilities of violating a constraint on the first and subsequent attempts were averaged over all students. The X-axis represents the attempt number (first, second and so on) when a student violated a constraint. The Y-axis shows the probability of violating these constraints. The probability of making a mistake is initially higher for the experimental group than the control group even

[3] Effect size = (Experimental Mean – Control Mean) /Standard Deviation of both groups

though not significant. Figure 4 indicates that both groups learnt the constraints in a similar manner. Both learning curves have a good fit to the power curve, indicating that the transferability of learning is high for both groups

We also investigated the number of constraints learnt by both groups. For each constraint in a student model, the first 5 attempts and the last 5 attempts during which a constraint was relevant was considered. If the probability of violating a constraint was reduced by 0.7 during the last 5 attempts, then that constraint was considered to be learnt. This analysis revealed that the experimental group learnt a significantly higher number of constraints than the control group (2.3 vs 1.2 p= 0.02).

2.2 Subjective Responses

Table 2 presents the subjective responses about various aspects of the dialogues. The starting and the ending points of the Likert scale had descriptive labels and the middle points had only numeric labels. For instance, when asking about overall quality of the dialogues, the starting and the ending labels were Poor (1) and Excellent (5) The points 2, 3 and 4 were indicated on the scale. The impression about the quality of the dialogues and the ease of understanding the questions were similar between the groups. However there was clear evidence that the control group did not like having to go through the entire dialogue.

Table 2. Subjective responses about tutorial dialogues (standard deviation reported in parentheses)

Question	Likert scale	Control	Experimental	p
Overall quality of the dialogues	Poor to Excellent (1 to 5)	3.5 (1.0)	3.7(0.8)	ns
Length of the dialogues	Too long to Too short (1 to 5)	2.6 (0.9)	3.2(0.5)	0.002
Ease of understanding the questions	Very Hard to very easy (1 to 5)	3.1(1.0)	3.4(0.8)	ns

3. Conclusions

We discuss an evaluation study that investigates the effectiveness of adaptive tutorial dialogues in EER-Tutor. The control group participants received non-adaptive dialogues regardless of their knowledge level and explanation skills. The experimental group participants received adaptive dialogues that were customised based on their student model. The study was conducted in their regular lab sessions and was limited to a single 2-hour session. At the end of the session the performance of the experimental group participants increased significantly more than their peers with an effect size of 0.69. The experimental group also learnt a significantly higher number of constraints than the control group. These results strongly suggest that the adaptive dialogues had a positive effect on learning database design. These results are significant because (i) the difference between the two groups was minimal (i.e. the only difference was the adaptivity of the dialogues) and (ii) the duration of the study was limited to a single 2- hour session. The subjective responses indicated no difference in their impression towards the quality of the dialogues and/or the understandability of the questions. However there was clear evidence that the control group did not like having to go through the entire dialogue before resuming problem-solving.

The participants were given the opportunity to interact with the system after this study. These interactions will be analysed to see how motivated they were to use EER-Tutor in their own time. Also we plan to use performance on their assignment which requires them to design a complex data model as a delayed post-test to investigate their improvement in their knowledge in database design.

Acknowledgements:

We would like to thank Benedict du Boulay from University of Sussex for his help with the evaluation study.

References

[1] Aleven, V., Ogan, A. Popescu, O. Torrey, C., & Koedinger, K. R. (2004). Evaluating the effectiveness of a Tutorial Dialogue System for Self-Explanation, Lester, J. C., Vicario, R.M., Papaguacu, F. (eds.) *Proceedings of ITS2004*, (pp 443-454). Alagoas, Brazil: Springer-Verlag.

[2] Bloom, B. (1984) The 2-sigma problem: The search of group instruction as effective as one-to-one tutoring, *Educational Researcher* 13, 3-16.

[3] Grasser, A.C., VanLehn, K., Rose, C.P., Jordan P. W., & Harter, D. (2001) Intelligent Tutoring Systems with Conversational Dialogue *AI magazine*, 22(4), 39-51.

[4] Millis, B., Evens, M., & Freedman, R. (2004). Implementing Directed Lines of Reasoning in an Intelligent Tutoring System using the Atlas Planning Environment, *Proceedings of the International Conference on Information Technology*: *ITCC 2004*, (pp. 729-733). Nevada-USA: IEEE Computer Society.

[5] Mitrovic, A., Martin, B. & Suraweera, P. (2007). Intelligent Tutors for all: Constraint-based modeling methodology, systems and authoring, *IEEE Intelligent Systems*, 22(4), 38-45.

[6] Mitrovic, A., Martin, B., & Mayo, M. (2002) Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, 12 (2-3), 243-279.

[7] Weerasinghe, A., & Mitrovic, A. (2008). A Preliminary Study of a General Model for Supporting Tutorial Dialogues. *Proceedings of the International Conference in Computers in Education*, (pp. 125-132). Taipei, Taiwan: Asia-Pacific Society for Computers in Education.

[8] Weerasinghe, A., & Mitrovic, A. (2006). Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based tutors, Aleven V. Ashley, K., Lynch, C. and Pinkwart, N. (eds.), Workshop on Intelligent Tutoring Systems for ill-defined domains at ITS2006, (pp. 55-64). Jhongli, Taiwan.

[9] Weerasinghe, A., & Mitrovic, A. (2006). Facilitating Deep Learning through Self-Explanation in an Open-ended Domain, *Knowledge-based and Intelligent Tutoring Systems* 10(1), 3-19.

Evaluating a General Model of Adaptive Tutorial Dialogues

Amali Weerasinghe[1], Antonija Mitrovic[1], David Thomson[1],
Pavle Mogin[2], Brent Martin[1]

[1]*Intelligent Computer Tutoring Group, University of Canterbury, New Zealand*
[2]*Victoria University of Wellington, Wellington New Zealand*
{amali.weerasinghe, david.thomson}@pg.canterbury.ac.nz, pavle.mogin@ecs.vuw.ac.nz,
{tanja.mitrovic, brent.martin}@canterbury.ac.nz

Abstract: Tutorial dialogues are considered as one of the critical factors contributing to the effectiveness of human one-on-one tutoring. We discuss how we evaluated the effectiveness of a general model of adaptive tutorial dialogues in both an ill-defined and a well-defined task. The first study involved dialogues in database design, an ill-defined task. The control group participants received non-adaptive dialogues regardless of their knowledge level and explanation skills. The experimental group participants received adaptive dialogues that were customised based on their student models. The performance on pre- and post-tests indicate that the experimental group participants learned significantly more than their peers. The second study involved dialogues in data normalization, a well-defined task. The performance of the experimental group increased significantly between pre- and post-test, while the improvement of the control group was not significant. The studies show that the model is applicable to both ill- and well-defined tasks, and that they support learning effectively.

Keywords: adaptive tutorial dialogues, constraint-based tutors, Ill-defined tasks, well-defined tasks

1. Introduction

One of the aspirations of AIED research is to explore how intelligent systems can achieve the same effectiveness as in human one-on-one tutoring. One of the major factors contributing to the effectiveness of human tutors is the conversational aspect of instruction. Dialogues provide opportunities for students to reflect on their existing knowledge and to construct new knowledge. Some of the existing dialogue-based tutoring systems are Why2-Atlas [1], Auto Tutor [2], CIRCSIM-Tutor [3], Geometry Explanation Tutor [4] and KERMIT-SE [5]. Why2-Atlas and Auto Tutor use dialogues as the main learning activity, while the others provide problem-solving as the main activity and use tutorial dialogues as a way of remediating student errors. For example, CIRCSIM-Tutor is a natural language tutor that helps students learn cardiovascular physiology related to regulation of blood pressure. The Geometry Explanation Tutor requires students to justify the problem-solving steps in their own words. KERMIT-SE, a database design tutor, engages students in dialogues when their solutions are erroneous. All these tasks except database design are well-defined: problem solving is well-structured, and therefore explanations expected from learners can be clearly

defined. In contrast, database design is an ill-defined task: the final result is defined only in abstract terms, and there is no algorithm to find it [6].

Our goal is to develop a general model for supporting dialogues across domains. Based on the findings of two Wizard-of-Oz studies [7], we developed a model consisting of three parts: an error hierarchy, tutorial dialogues and rules for adapting them. The error hierarchy categorizes all error types in a domain. At the leaf level, an error type is associated with one or more violated constraints. (The knowledge bases of our constraint-based tutors are represented in terms of constraints.) The error types are then grouped into higher-level categories. Remediation is facilitated through tutorial dialogues, one of which is developed for each error type. When there are multiple errors in a student solution, the hierarchy is traversed to select the error most suitable for discussion and the corresponding dialogue is then initiated. Finally, the adaptation rules are used to individualize the dialogues to suit the student's knowledge and reasoning skills by controlling their timing and the exact content. In response to the generated dialogue learners are able to provide answers by selecting an option from a list. For a detailed discussion of the model see [7].

In this paper we discuss how we evaluated the effectiveness of our model supporting an ill-defined and a well-defined task. The first study investigated the effectiveness of our model in database design (an ill-defined task), in the context of EER-Tutor [8]. In database design, students design database schemas using the EER model. Students need to know the concepts of the EER data model, use world knowledge about different real-world scenarios (i.e. enrolling students in a university etc.) and be able to handle the ill-definedness of the task. In the second study, we evaluated our model in data normalization, using NORMIT [8]. Data normalization is the process of refining a relational database schema in order to ensure that all relations are of high quality. This task requires normalizing a given database schema using the specified procedure. NORMIT contains a page for each step of this procedure, and students are requested to complete one step before continuing with the next one. The following two sections present the results of the study, followed by discussion and conclusions.

2. EER-Tutor Study

We conducted a study with the EER-Tutor in March 2010 at the University of Canterbury, which involved volunteers from an introductory database course. The objective of the study was to investigate whether adaptive dialogues are more effective in improving learning than non-adaptive dialogues in database design.

The participants were randomly assigned to groups. The experimental group received adaptive dialogues, while the control group had non-adaptive dialogues. The differences between the two groups were in dialogue selection, dialogue prompts and additional support. Dialogues for the control group were selected using the depth-first traversal of the error hierarchy. The first violated constraint that was found in the traversal was selected for discussion. As the errors in the hierarchy are ordered from simpler to more complicated errors, the depth-first search results in the simplest error for the control group.

The dialogues in our model consist of four stages [7]: (i) a problem-independent prompt discusses the relevant domain concept for the selected error; (ii) a problem-dependent prompt discusses the error in the context of the current problem; (iii) a

corrective action prompt provides an opportunity to understand how to correct the error and (iv) a reinforcement prompt, providing another opportunity to learn the related domain concept. The control group saw the entire dialogue regardless of the number of times they have seen the dialogue previously or their responses to the dialogue prompts. As the result, the same solution submitted by two different students with different knowledge levels in the control group received identical dialogues. In contrast, an experimental group participant receives the problem-dependent prompt (prompt (ii)) the first time a mistake is done. If s/he makes this type of error repeatedly, the dialogue will start from the problem-independent prompt. The exit point of the dialogue for the experimental group is customized based on the student's past interactions with the dialogues. For a detailed description, see [7].

When an experimental group participant abandons a problem (i.e. changes a problem without attempting it) or has been inactive for a period of time, they were asked whether they needed help. If they requested help then their solution was evaluated and an error was selected for discussion based on their student model. The control group did not receive this support.

The study consisted of four stages: pre-test, interactions with EER-Tutor, post-test and questionnaire. The pre- and post-tests had 6 questions each, of similar difficulty. We wanted to evaluate whether students' problem-solving abilities as well as explanation skills improved after interacting with the system. One question asked the participants to provide the database schema for the given requirements. This is a typical question that can be found in examinations, text books etc. The other three questions were aimed to understand the effect the system had on students' explanation skills.

The participants used EER-Tutor for the first time in their regular lab sessions during the third week of the course, for a single 2-hour session. At the beginning of the session students were given about 10 minutes to complete the pre-test, after which they interacted with the system. Towards the end of the session, they were given 10 minutes to complete the post-test and 5 minutes to answer a questionnaire.

Out of 104 students enrolled in the course, 77 participated in the study. There was no significant difference in the pre-test performance between the control and the experimental groups. Some students have not completed the post-test. Table 1 reports some statistics about the 65 participants who completed both pre-and post-tests.

Table 1. Some statistics from the EER-Tutor study (sd given in parentheses)

	Control (34)	Experimental (31)	p
Pre-test (%)	54.5 (18.1)	51.3 (16.1)	ns
Post-test mean (%)	61.2 (14.9)	69.9 (11.5)	0.005
Gain	6.8(15.6)	18.6 (16.8)	0.002
Normalised gain	0.002 (0.7)	0.3 (0.4)	0.01
Interaction time (min)	62.8 (22.1)	62.9 (24.1)	ns
Attempted Problems	8.6(4.8)	10.6(4.8)	ns
Solved problems	9.0(4.8)	7.9 (4.7)	ns
Total Dialogues received	12.1 (7.3)	14.0 (8.3)	ns
Questions answered	34.4 (25)	23.6 (14.6)	0.01
% of correct answers	61.4 (23.1)	59 (16.9)	ns

There were 31 participants in the experimental group and 34 in the control group, with no significant difference on the pre-test performances. The post-test performance of the experimental group was significantly better compared to their peers who received non-adaptive dialogues. Both the learning gain (post-test score – pre-test

score) and the normalised learning gain[1] of the group who received adaptive dialogues was also significantly higher than the gains of the control group.

There were no differences between the times spent with the system, the numbers of attempted and solved problems, and the number of dialogues received. The control group answered a significantly higher number of questions than their peers. This was expected, as the control group had to go through the entire dialogue before resuming problem-solving. However, percentages of correct answers are similar for both groups.

The effect size[2] (Cohen's d) for learning gains of the two groups is 0.69 (the effect size based on the normalized gain is 0.51). The effect size obtained here is remarkable because the only difference between the two groups was the adaptivity of the dialogues.

In order to investigate how the students learnt the database design concepts in terms of constraints, we analyzed how frequently constraints were violated. Figure 1 illustrates the learning curves for both groups. The probabilities of violating a constraint on the first and subsequent attempts were averaged over all students. The x-axis represents the attempt number (first, second and so on) when a student violated a constraint. The y-axis shows the probability of violating these constraints. The probability of making a mistake is initially higher for the experimental group than the control group even though not significantly. Figure 1 indicates that both groups learnt the constraints in a similar manner.

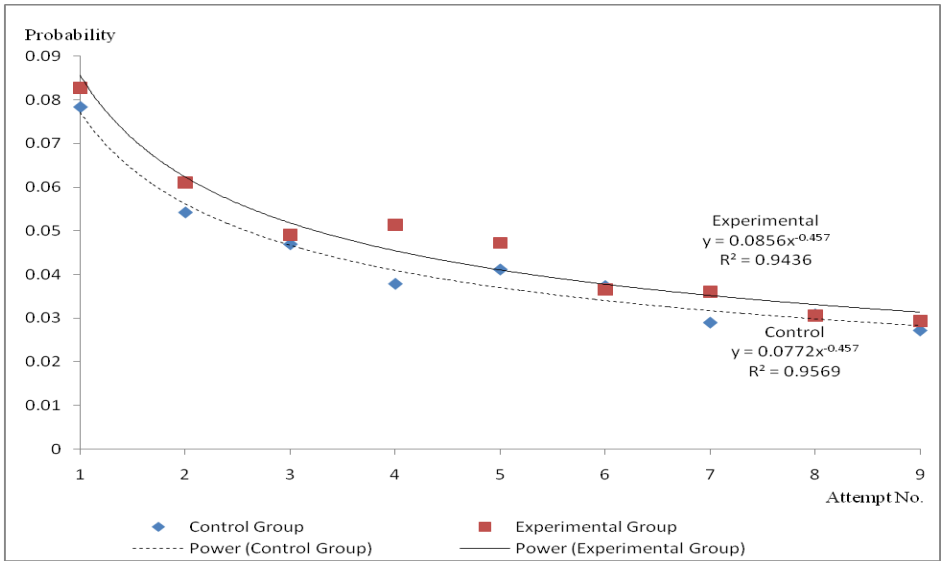


Fig 1: Probability of constraint violations – EER-Tutor study

We also investigated the number of constraints learnt by both groups. We used the first five attempts and the last attempts on each constraint to decide whether the status of the constraint changed from 'not known' to 'learnt' for a given student. If the probability of violating a constraint is below a pre-defined threshold then the constraint

[1] Normalised learning gain =learning gain/(1-pre-test score)

[2] Effect size = (Experimental Mean – Control Mean) /Standard Deviation of both groups

was deemed not known. Similarly, if the probability of violating a constraint is above the same pre-defined threshold then it was considered to be learnt. This analysis revealed that the experimental group learnt a significantly higher number of constraints than the control group (2.3 vs 1.2, p= 0.02).

Table 2 presents the subjective responses about various aspects of the dialogues. The impression about the quality of the dialogues and the ease of understanding the questions were similar between the groups. However there was clear evidence that the control group did not like having to go through the entire dialogue.

Table 2. Subjective responses about tutorial dialogues (sd given in parentheses)

Question	Likert scale	Control	Experimental	p
Quality of the dialogues	Poor to Excellent (1 to 5)	3.5 (1.0)	3. 7(0.8)	ns
Length of the dialogues	Too long to Too short (1 to 5)	2.6 (0.9)	3.2 (0.5)	0.002
Ease of understanding the questions	Very Hard to Very Easy (1 to 5)	3. (1.0)	3.4 (0.8)	ns

3. NORMIT Study

We conducted a study with NORMIT in September 2010 at the Victoria University of Wellington, which involved 20 volunteers from a database system engineering course in a single, 1-hour session. The objective and the experimental setup for this study are similar to that of EER-Tutor study. Pre-and the post-tests were designed to explore the system's effect on both the students' problem-solving abilities and explanation skills. Both pre- and post-tests had 4 questions each, of similar difficulty. Two questions requested students to solve very simple problems, and explain their solutions. The other two questions requested students to specify definitions of concepts. Some students have not completed the post-test. Table 3 reports some statistics about the 18 participants who completed both tests. Each group had 9 students.

Table 3: Some statistics from the NORMIT study (sd given in parentheses)

	Control (9)	Experimental (9)	p
Pre-test (%)	68.1 (30.0)	69.4 (29.4)	ns
Post-test (%)	72.2 (24.0)	86.1(15.9)	ns
Gain	4.2 (32.4)	16.7 (27.2)	ns
Interaction time (min)	60.1(24.7)	47.7 (16.8)	ns
Attempted Problems	7.1 (3.0)	5.9 (2.1)	ns
Solved problems	6.1 (3.0)	5.4 (2.0)	ns
Total Dialogues received	27.8 (14.6)	23.6 (11.3)	ns
Questions answered	55.7 (37.4)	23.9 (11.5)	0.01
% of correct answers	6.9 (4.1)	8.2 (4.7)	ns

There were no significant differences between the pre-test and post-test performances of the two groups, as well as between the gains. The performance of the experimental group increased significantly between pre- and post-test (paired t-test, t=1.84, p=0.052), while the improvement of the control group was not significant. The effect size for learning gains of the two groups is 0.4.

As the study was limited to a single lab session, the two groups spent a similar time interacting with the system. The groups attempted and solved a similar number of problems, and received a similar number of dialogues.

The control group participants answered significantly more questions than their peers, as was the case in the EER-Tutor study. This can be expected as the control group had to go through the entire dialogue every time a dialogue is given to the student. However, percentages of correct answers are similar for both groups.

Figure 2 presents the learning curves for both groups. The probability of making a mistake is initially higher for the experimental group than the control group even though not significantly. The learning curves indicate that the learning rate of the experimental group is higher than that of the control group. Similar to the EER-Tutor study, we also investigated the number of constraints learnt by both groups. There was no significant difference between the numbers of constraints learnt.

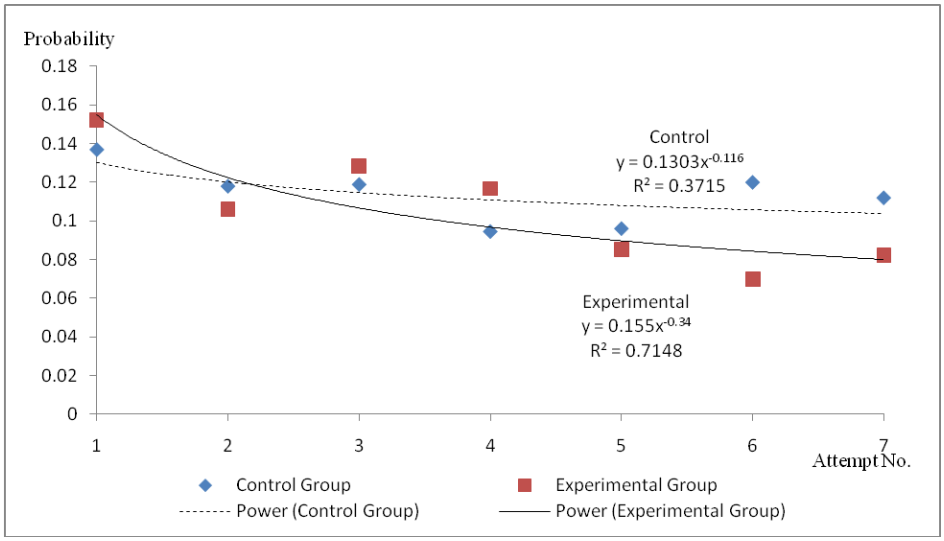


Fig 2: Probability of constraint violations – NORMIT study

We also explored the users' impressions about various aspects of tutorial dialogues using questionnaires (Table 4). The questions used for the EER-Tutor study were used here. The impression about the quality of the dialogues and the ease of understanding the questions were similar between the groups. Unlike the EER-Tutor study, there was no evidence from the control group that the non-adaptive dialogues were too long.

Table 4. Subjective responses about tutorial dialogues (sd given in parentheses)

Question	Likert scale	Control	Experimental	p
Quality of the dialogues	Poor to Excellent (1 to 5)	3.3 (0.5)	3.1(1.0)	ns
Length of the dialogues	Too long to Too short (1 to 5)	3.1 (0.8)	3.3(0.5)	ns
Ease of understanding the questions	Very Hard to Very Easy (1 to 5)	3.4(0.7)	3.1(0.7)	ns

4. Discussion and Conclusions

We presented how we evaluated the effectiveness of our model for supporting tutorial dialogues in two very different tasks. Our model facilitates adaptive dialogues based on a student's knowledge and their interaction with the dialogues. The dialogues discuss a student's mistake in the current context and the relevant domain concepts.

In EER-Tutor study the learning gain of the experimental group (that received adaptive dialogues) is significantly higher than the gain of their peers, with the effect size of 0.69. The experimental group also learnt a significantly higher number of constraints. These results strongly suggest that adaptive dialogues had a positive effect on learning database design. This is a significant result because (i) the difference between the two groups was minimal (i.e. the only difference was the adaptivity of the dialogues) and (ii) the study was limited to a single 2- hour session.

In the NORMIT study, there were no significant differences between the pre-test and post-test performances of the two groups, as well as between the gains. This might be due to the small number of participants (20 vs 65 in EER-Tutor study). However, we can observe similar trends in learning in both studies: significantly higher number of constraints learnt in EER-Tutor study, and a higher learning rate in NORMIT study by the respective experimental groups compared to their peers.

In both studies we used dialogues to discuss the errors in the problem-solving process, and not as the main activity to learn the domain knowledge. The task facilitated in EER-Tutor requires world knowledge about different real-world scenarios such as enrolling students in a university, or customers interacting with a bank. In the EER-Tutor study, the model was used to support dialogues in an ill-defined task with the well-defined domain theory. In the NORMIT study, dialogues facilitated learning a well-defined task with the well-defined domain theory. Therefore, our model has shown evidence of enhancing learning of a domain in the WDIT quadrant (well-defined domain, ill-defined task) and WDWT quadrant (well-defined domain, well-defined task) [6]. As the next step, we plan to explore the possibility of developing the model for a task such as essay writing or legal argumentation in the IDIT quadrant (Ill-defined domain, Ill-defined task).

The three highest levels of the error-hierarchy (the first component of the model) are domain-independent. The top level node is *All Errors*, which is then further divided into *Basic Syntax Errors* and *Errors dealing with the main problem solving activity*. The latter is further divided into (i) *Using an incorrect solution component type*, (ii) *Extra solution components,* (iii) *Missing solution components*, (iv) *Associations* and (v) *Failure to complete related changes*. Further divisions of these nodes and the node *Basic Syntax Errors* deal with domain-specific concepts. Even though tutorial dialogues consist of domain-specific prompts, the structure is domain-independent. Adaptation rules (the last component) which customise dialogue prompts are domain-independent except for the time period of inactivity the tutor waits before intervening.

We also investigated whether our model can be used in other domains. We tried to fit the errors from two different domains: logical database design and fraction addition into our model. Logical database design involves mapping high-level, conceptual ER schemas to relational schemas using the 7-step mapping algorithm [9]. We used the constraint-base of ERM-Tutor [10], a constraint-based tutor for teaching logical database design and developed the error hierarchy categorizing all the constraints. Then we explored whether we could develop dialogues for each type of error. All these were done on paper and the model could be developed for logical database design. We

repeated the steps of (i) developing the error hierarchy using the constraints developed for fraction addition and (ii) developing dialogues for each type of error. The outcome of our attempt is a model that could be implemented to support dialogues in fraction addition. Therefore we have developed models for four different domains: (i) database design (ii) data normalization (iii) logical database design and (iv) fraction addition. The first two were implemented and evaluations indicate that the model can enhance learning the domain knowledge. The last two were done on paper and our attempt provides evidence that the model can be used in different domains.

For a newly created constraint-based tutor, developing our model to support dialogues involves (i) developing the error hierarchy to categorize the errors in the domain using the constraint-base (ii) designing the dialogues for each type of error and (iii) customizing the domain-dependent features (i.e. inactive time period) in the adaptation rules. Furthermore, even though this model was developed for constraint-based tutors, it can be used in any ITS with a problem-solving environment. In such ITSs, a student solution is evaluated and feedback is provided on errors regardless of the mechanism/methodology used for diagnosis. Therefore, the error hierarchy (the first component of the model) could be developed using the error types of that domain. Tutorial dialogues (the second component of the model) need to be written for each type of error based on the dialogue structure. The third component of the model, rules for adapting dialogues, are domain independent (except for the inactive time period), and can be used across domains.

The future work includes conducting a larger NORMIT study and exploring the possibility of developing a model for an ill-defined task in an ill-defined domain.

References

1. VanLehn, K., Graesser, A. C., Jackson, G. T., Jordan, P., Olney, A., & Rose, C. P. :When are tutorial dialogues more effective than reading? Cognitive Science 31(1), 3-52, (2007).
2. Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A., et al.: AutoTutor: A tutor with dialogue in natural language. Behavioral Research Methods, Instruments and Computers, *36,* 180–193,(2004).
3. Evens, M. and Michael, J., One-on-One Tutoring By Humans and Computers. Mahwah, New Jersey: Lawrence Erlbaum Associates, 2006.
4. Aleven, V., Ogan, A. Popescu, O. Torrey, C., Koedinger, K.: Evaluating the Effectiveness of a Tutorial Dialogue System for Self-Explanation. In: Lester, J. et al. (Eds.) ITS2004, LNCS, vol. 3220, pp 443-454, Springer-Verlag, Berlin (2004).
5. Weerasinghe, A., Mitrovic, A.: Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. Knowledge-based and Intelligent Tutoring Systems 10(1), 3-19 (2006).
6. Mitrovic, A., Weerasinghe, A.: Revisiting the Ill-Definedness and Consequences for ITSs. Dimitrova, V. et al. (Eds.) Proc. Artificial Intelligence in Education, Frontiers in Artificial Intelligence and Applications, vol. 200, pp. 375-382 (2009).
7. Weerasinghe, A., Mitrovic, A., Martin, B.: Towards Individualized Dialogue Support for Ill-Defined Domains IJAIED, Special Issue on Ill-Defined Domains, 19(4): pp. 357-379 (2009).
8. Mitrovic, A., Martin, B., Suraweera, P: Intelligent Tutors for All: Constraint-based Modeling Methodology, Systems and Authoring. IEEE Intelligent Systems 22(4), 38-45 (2007).
9. Elmasri, R., Navathe, S., Fundamentals of Database Systems (5th ed.). Boston: Addison-Wesley (2007).
10. Milik, N., Marshall, M., Mitrovic, A.: Teaching logical database design in ERM-Tutor.: Ikeda, M., Ashley, K. (Eds.) Proc. of ITS2006, pp. 707-709 (2006).

Facilitating Adaptive Dialogues in EER-Tutor

Amali Weerasinghe, Antonija Mitrovic

In this interactive event, the participants will have the opportunity to solve problems in EER-Tutor and engage in dialogues. They will be able to experience how the dialogues are customised based on their knowledge level and their interactions with the dialogues. We demonstrate how the content of the dialogues are adapted by customisng the starting and the finishing points of dialogues based on the student model.

URL for EER-Tutor: http://ictg.cosc.canterbury.ac.nz:8005/

1. Logging into the system

Usercodes and passwords will be provided at the interactive event of the AIED2011 conference. You can also ask for a user code and a password by emailing amali.weerasinghe@pg.canterbury.ac.nz

Please make sure that you tick the first-time login box, if you are a first time user.

2. Working space in EER-Tutor

Fig. 1 presents the working space after you logged into the system. Users are given a real-world scenario in natural language and asked to design the data model using the ER modelling notation.

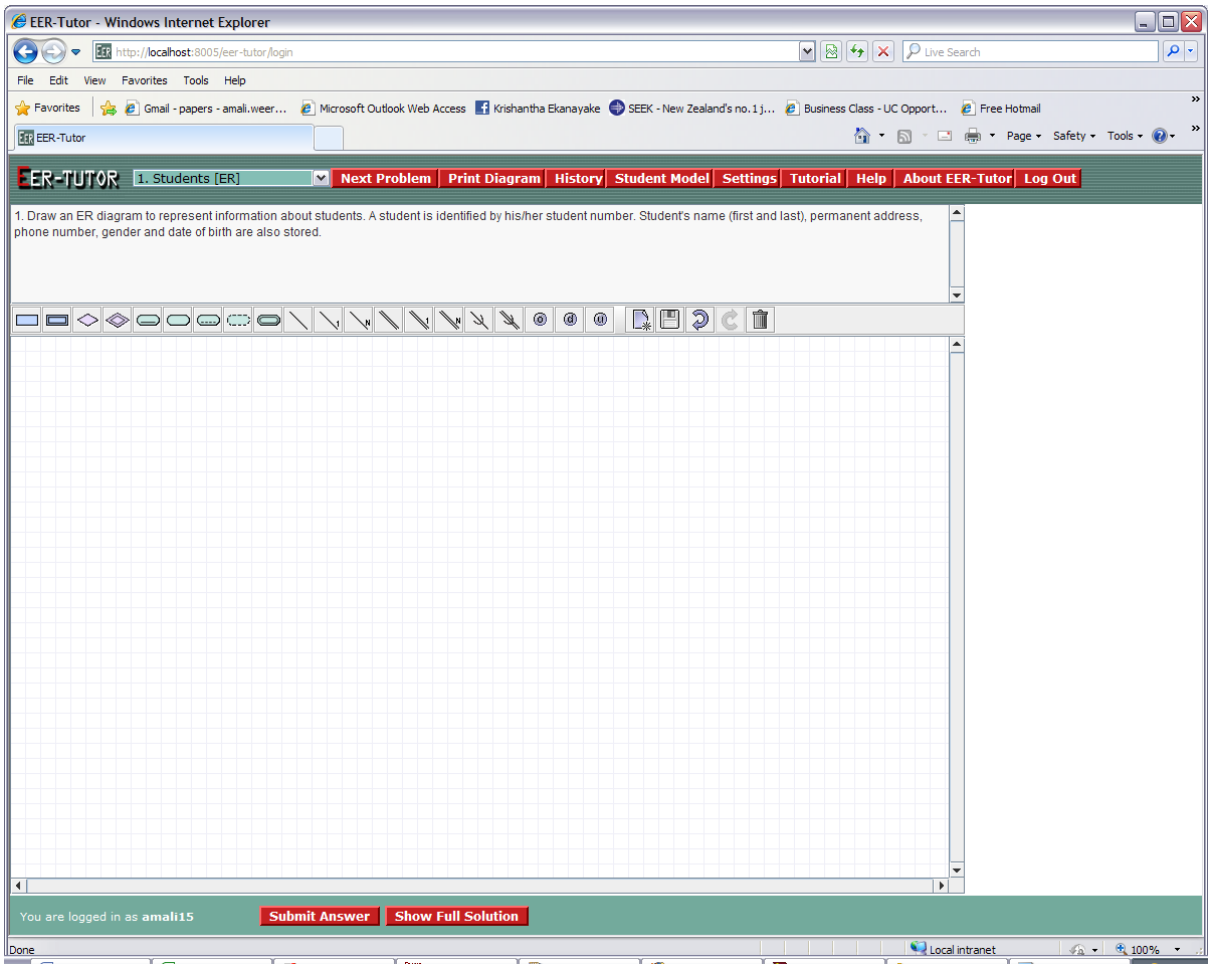


Fig.1: Working space of EER-Tutor

3. Developing a data model in EER-Tutor

To include a symbol with a text label in the data model, highlight the phrase in the problem text and then select the symbol required and then click on the working space. For example, to include STUDENT as a regular entity, highlight the word *Student* in the problem text, click on left most symbol (rectangle represents a regular entity) and then on the problem space.

Tool tips are available for each symbol for easy identification. Online tutorial provides detailed instructions on how to use EER-Tutor.

4. Submitting a Solution

Change to problem 6 using the drop down menu on top left hand corner. You may want to submit the following solution to the system. It has several mistakes i.e CHAPTER should be modelled as a weak entity (CHAPTER is currently a regular entity), CONTAINS should be modelled as an identifying relationship (currently it is a regular relationship), CHAPTER needs a partial key attribute etc.

When this solution is submited, EER-Tutor will provide user the opportuntiy to enage in a dialogue (Fig.2). The objective of the dialogue here is to discuss why CHAPTER should not be

modelled as a regular entity. Even though there are several mistakes in the submitted solution, the error about the CHAPTER being a weak entity is selected based on the error hierarchy which categorises errors from simpler ones to more complicated ones and the probability of making a mistake based on the student model.

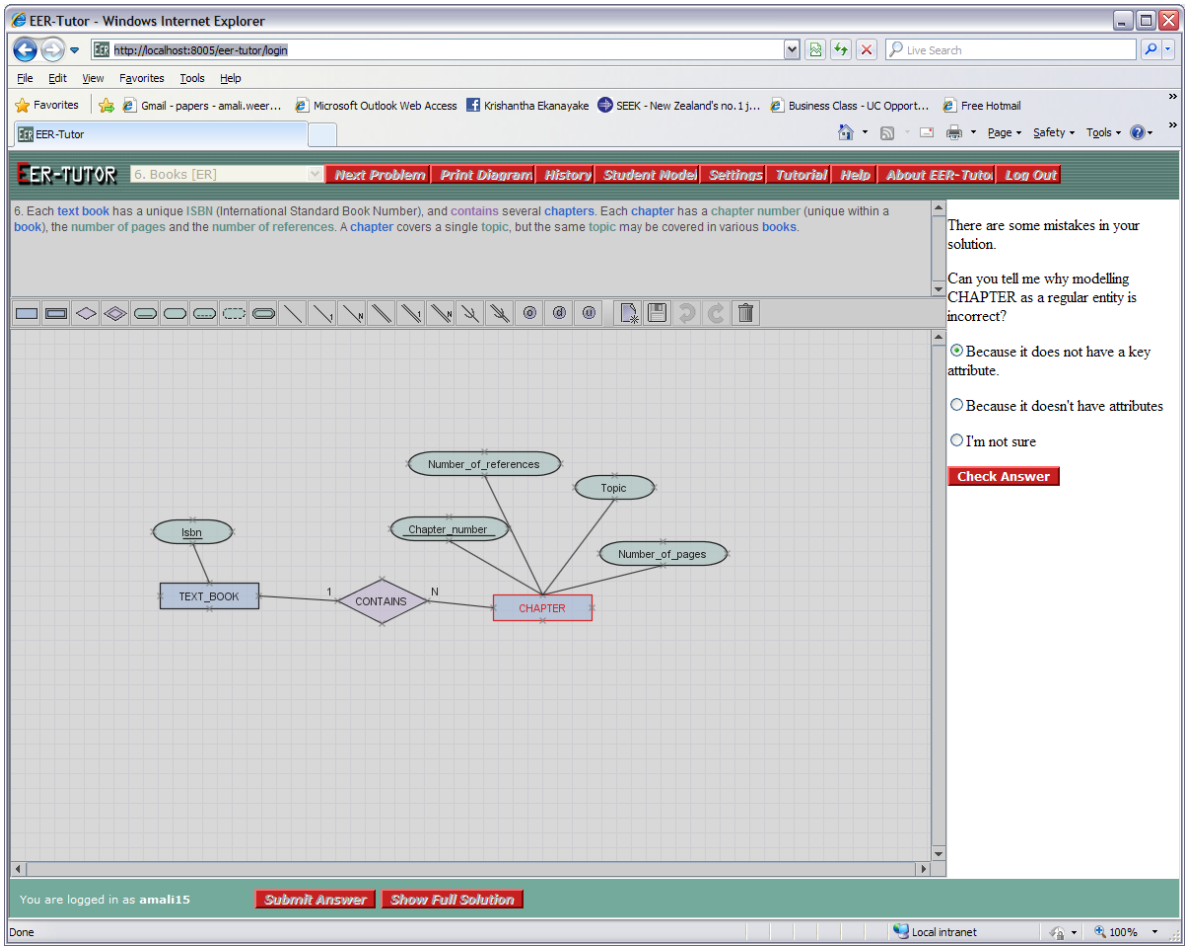


Fig.2 – Initiation of the dialogue

5. Behaviour of the diaogue when the user makes a mistake for the first time

As the error is done for the first time, the dialogue will discuss the error in the current problem context . This prompt is referred to as the problem-dependeent prompt.[1] This is the level 2 prompt of the dialogue. If a error is done repeatedly, then the dialogue will start disussing the relevant domain concept . This scenario will be discussed in section 7.

The correct answer here is "because it does not have a key attribute.". If the correct answer is selected, EER-Tutor will let the user resume problem solving without going through any more dialogue prompts (Fig. 3). We do not want the users to go through an entire dialogue because

[1] The dialogues consist of four stages: (i) a problem-independent prompt discusses the relevant domain concept for the selected error; (ii) a problem-dependent prompt discusses the error in the context of the current problem; (iii) a corrective action prompt provides an opportunity to understand how to correct the error and (iv) a reinforcement prompt, providing another opportunity to learn the related domain concept.

problem solving is the main activity and the dialogues are used to discuss errors in a submitted solution.

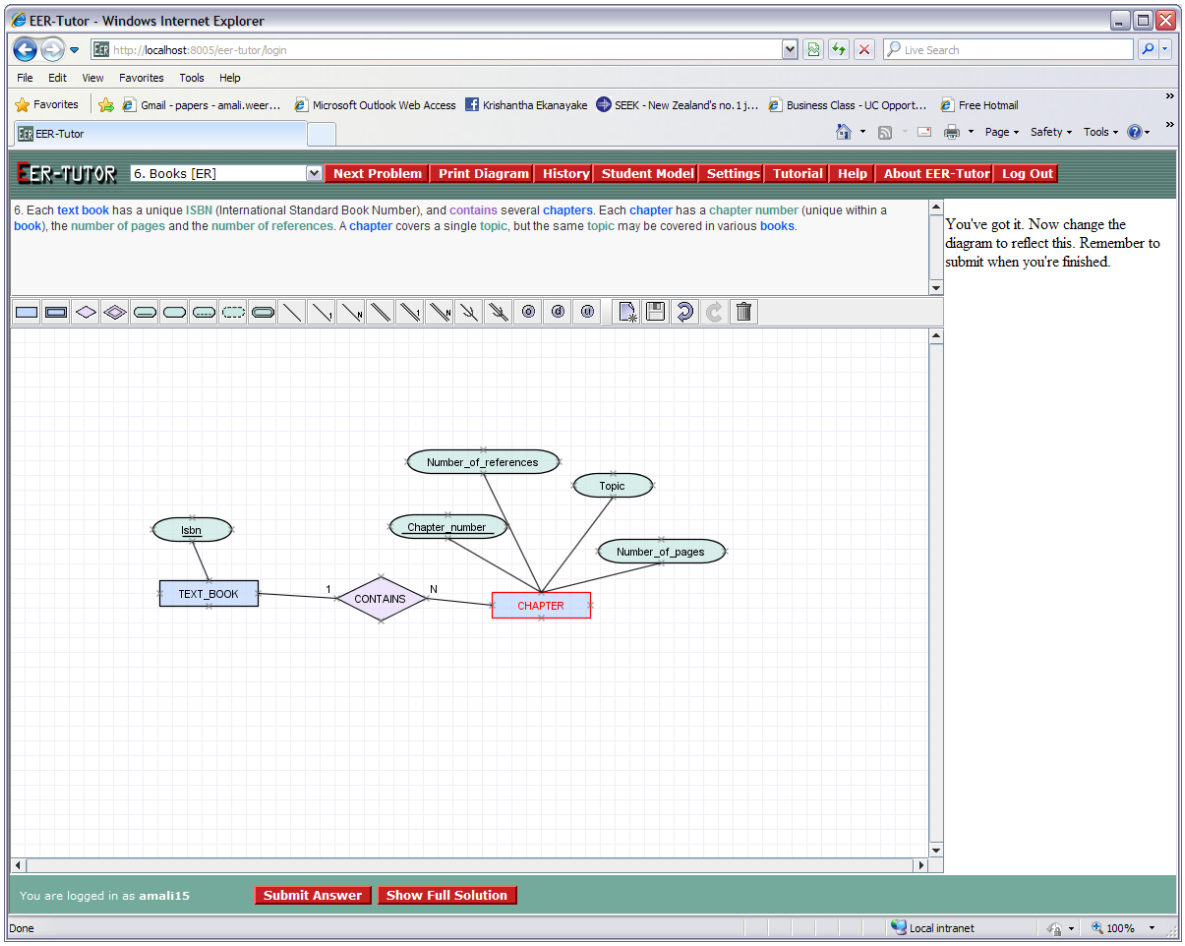


Fig.3: End of the current dialogue

6. Adaptation of the dialogue when the user makes the same mistake again

Submit the same solution again without any changes (Fig. 2). Now this is the second time the same mistake is made. A repetition of the same mistake indicates that the user has not fully understood the discussion provided by the dialogue. As a result, the user is encouraged to go through one extra level of the dialogue even if the user answers correctly to the first prompt. i.e. The user will see level 3 of the dialogue which prompts the user to specify how the error can be corrected. (Figures 2 ,4 and 5).

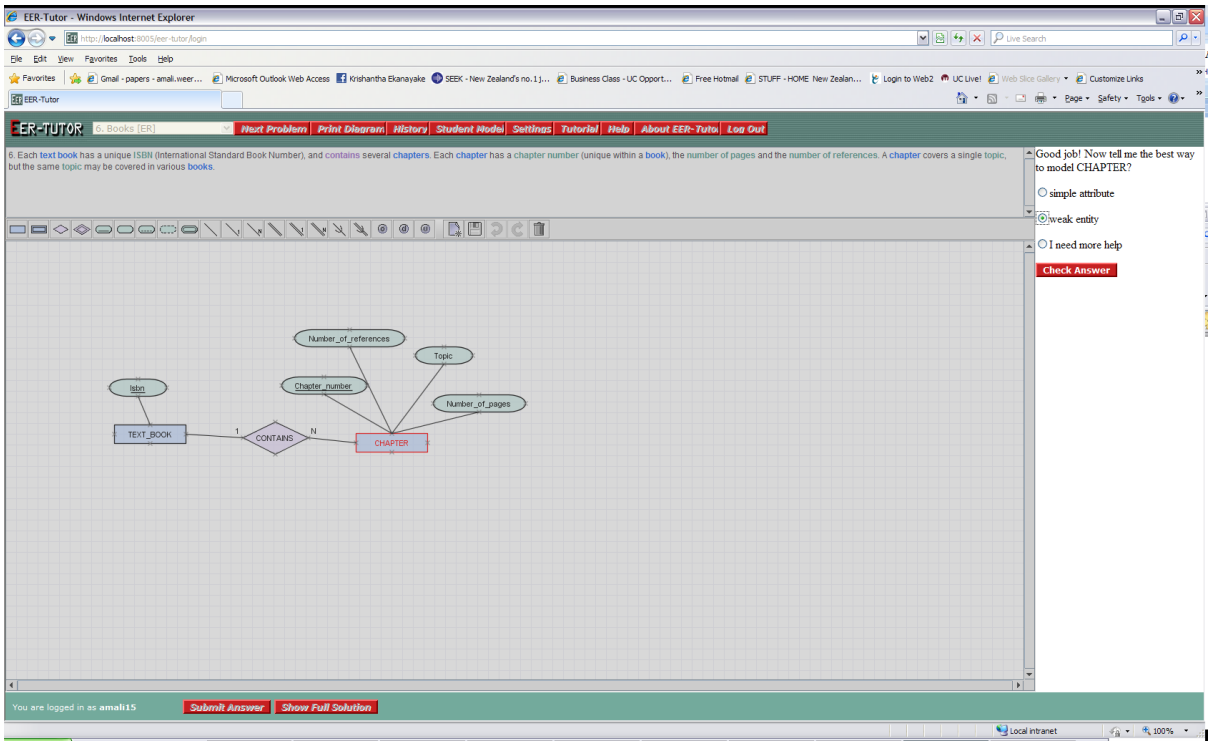


Fig. 4 – Providing a corrective action prompt - an opportunity to understand how to correct the error.

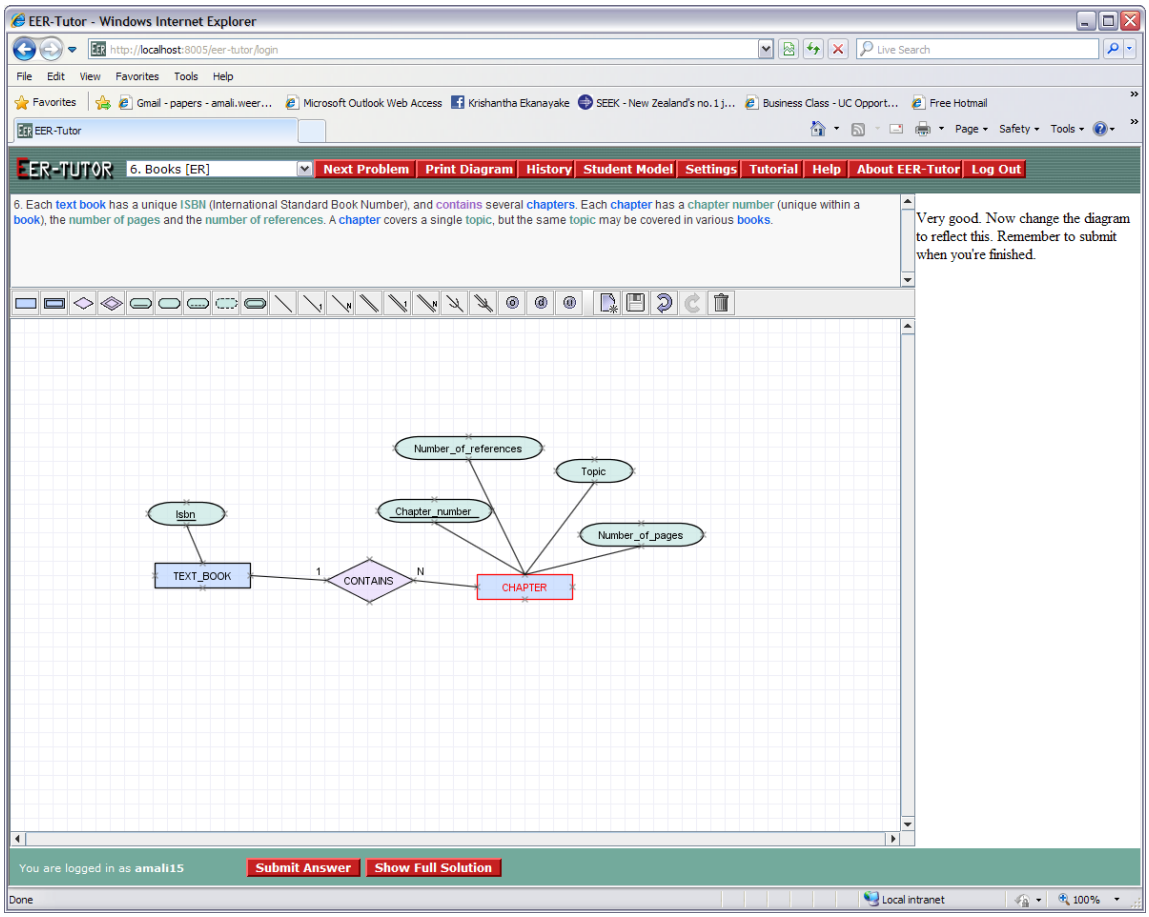


Fig. 5 – End of the dialogue

From this point onwards, the EER-Tutor will ask the user to go through levels two and three for this specific dialogue until the end of the current session (Figures 2, 3 and 4). The exit point of a dialogue depends on the accuracy of the current user response and the level of the dialogue he/she received in the previous instance for this mistake within the current session.

7. Adaptation of the dialogue when the user makes the same mistake repeatedly

Submit the same solution again without any changes (Fig. 2). Now this is the third time this mistake is made. An error done repeatedly indicates evidence of lack of relevant domain concepts. Therefore, in such a situation EER-Tutor will discuss the relevant domain concept first (Fig. 6) and then discuss the error in the current problem context (Fig. 2). The prompt that discusses the corresponding domain concept is known as the problem independent prompt. This is the top-level prompt of the dialogue.

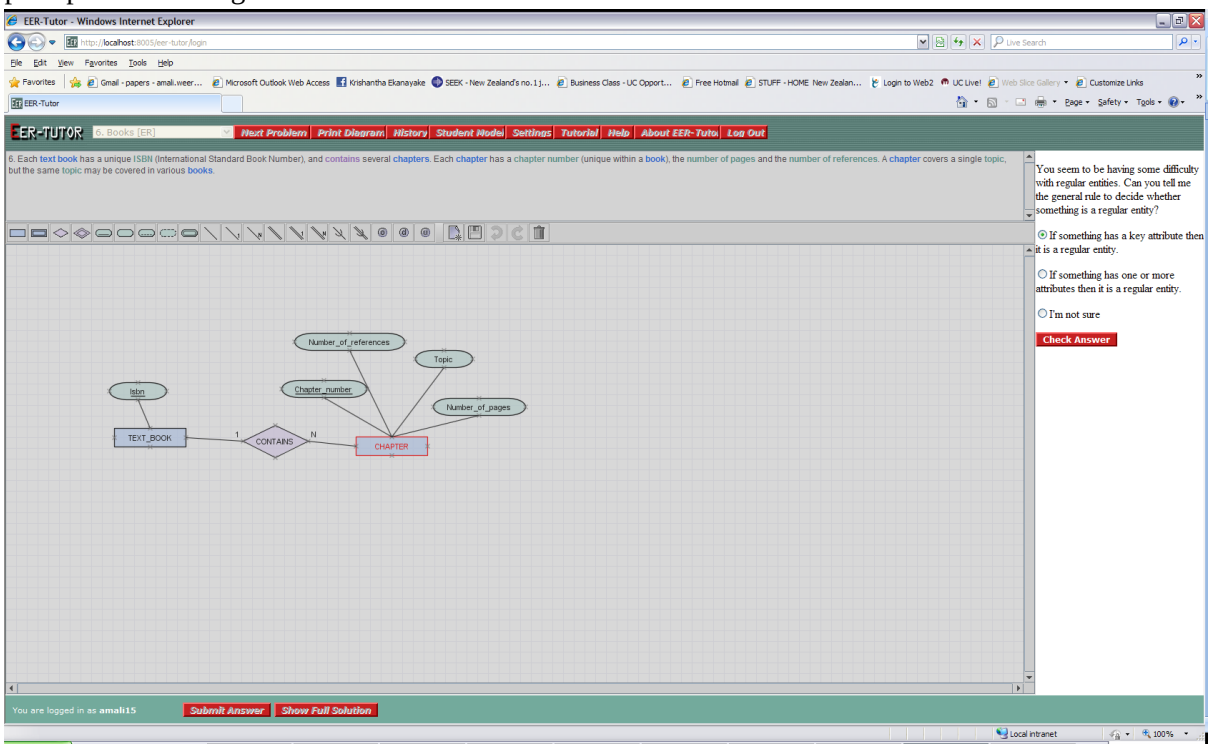


Fig.6 - Providing a problem-independent prompt: an opportunity to understand the relevant domain concept

8. Adaptation of the dialogue when the user makes the same mistake repeatedly (even after three instances in the current session)

Submit the solution again without any changes. This is the fourth time this mistake is made. Now the student will continue to see the level1 prompt (problem independent prompt) of this specific dialogue until the end of the session. The exit point of this dialogue depends on the accuracy of the user response and the level of the dialogue he/she received in the previous instance for this mistake.

EER-Tutor maintains the user interactions with the dialogues for each error type and uses this information to customise the dialogues for each student.

E.2 Co-author ship Forms

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 3.3 is extracted from the paper "Weerasinghe, A. and Mitrovic, A. (2006) Studying human tutors to facilitate self-explanation. Jhongli, Taiwan: 8th International Conference on Intelligent Tutoring Systems, 26-30 Jun 2006. In Lecture Notes in Computer Science (LNCS) 4053(Intelligent Tutoring Systems) 713-715".

Please detail the nature and extent (%) of contribution by the PhD candidate:

This was my own research. I conducted the study and analysed results. Prof. Mitrovic provided advice. The paper was reviewed by Prof. Mitrovic. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all
The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature: Atmitrovic Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

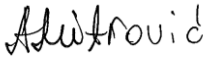
Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 3.1 is extracted from the paper :Weerasinghe, A. and Mitrovic, A. (2006) Individualizing Self-Explanation Support for Ill-Defined Tasks in Constraint-based Tutors. Jhongli, Taiwan: 8th International Conference on Intelligent Tutoring Systems, Workshop on Intelligent Tutoring Systems for Ill-Defined Domains, 27 Jun 2006. 56-64".

Please detail the nature and extent (%) of contribution by the PhD candidate:
The paper is based on Prof. Mitrovic's idea. I helped with the literature review.
My contribution is about 75%.

Certification by Co-authors:
If there is more than one co-author then a single co-author vcan sign on behalf of all
The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature:  Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 3.3 was extracted from the paper "Weerasinghe, A., Mitrovic, A. and Martin, B. (2007) Towards a General Model for Supporting Explanations to Enhance Learning. Los Angeles, CA, USA: 13th International Conference on Artificial Intelligence in Education (AIED 2007), 9-13 Jul 2007. In Frontiers in Artifical Intelligence and Applications 158 665-667."

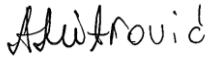
Please detail the nature and extent (%) of contribution by the PhD candidate:

This was my own research. Prof. Mitrovic provided advice. The paper was reviewed by Prof. Mitrovic. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all
The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature:  Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 3.7 is extracted from the paper " Weerasinghe, A. and Mitrovic, A. (2008) A Preliminary Study of a General Model for Supporting Tutorial Dialogues. Taipei, Taiwan: 16th International Conference on Computers in Education (ICCE2008), 27-31 Nov 2008. 125-132.

Please detail the nature and extent (%) of contribution by the PhD candidate:

This was my own research. I conducted the study and analysed results. Prof. Mitrovic provided advice. The paper was reviewed by Prof. Mitrovic. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all
The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature: Atmitrovic Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Sections 3.1 to 3.7 are extracted from the paper "Weerasinghe, A., Mitrovic, A. and Martin, B. (2009) Towards Individualized Dialogue Support for Ill-Defined Domains. International Journal on Artificial Intelligence in Education 19(4): 357-379."

Please detail the nature and extent (%) of contribution by the PhD candidate:

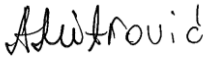
This was my own research. I conducted the evaluation studies presented in the paper. I also analysed the results. Prof. Mitrovic provided advice and reviewed the paper. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature:  Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 3.1 is extracted from the paper " Mitrovic, A. and Weerasinghe, A. (2009) Revisiting Ill-Definedness and the Consequences for ITSs. Brighton, UK: 14th International Conference on Artificial Intelligence in Education (AIED2009), 6-10 Jul 2009. In Frontiers in Artificial Intelligence and Applications 200, 375-382".

Please detail the nature and extent (%) of contribution by the PhD candidate:

The paper is based on Prof. Mitrovic's idea. I helped with the literature review. My contribution is about 40%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature: A Mitrovic Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 4.1 is extracted from the paper "Weerasinghe, A., Mitrovic, A., Van Zijl, M. and Martin, B. (2010) Evaluating the Effectiveness of Adaptive Tutorial Dialogues in database design. Putrajaya, Malaysia: 18th International Conference on Computers in Education ICCE 2010, 29 Nov-3 Dec 2010. In Proceedings of the 18th International Conference on Computers in Education 33-40."

Please detail the nature and extent (%) of contribution by the PhD candidate:

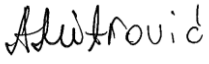
This was my own research. I conducted the evaluation study presented in the paper. I also analysed the results. Prof. Mitrovic provided advice and reviewed the paper. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature:  Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Sections 4.1 and 4.2 are extracted from the paper "Weerasinghe, A., Mitrovic, A., Thomson, D., Mogin, P. and Martin, B. (2011) Evaluating a General Model of Adaptive Tutorial Dialogues. Auckland, New Zealand: 15th International Conference on Artificial Intelligence in Education (AIED 2011), 28 Jun-1 Jul 2011. In Lecture Notes in Computer Science (LNCS): Artificial Intelligence in Education 6738, 394-402"

Please detail the nature and extent (%) of contribution by the PhD candidate:

This was my own research. I conducted the evaluation studies presented in the paper. I also analysed the results. Prof. Mitrovic provided advice and reviewed the paper. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature: Date: *2.10.2012*

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any PhD thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the PhD thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit (including electronic copy).

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Section 3.6 is extracted from the paper Weerasinghe, A. and Mitrovic, A. (2011) Facilitating Adaptive Tutorial Dialogues in EER-Tutor. Auckland, New Zealand: 15th International Conference on Artificial Intelligence in Education (AIED 2011), 28 Jun-1 Jul 2011. In Lecture Notes in Artificial Intelligence 630-631.

Please detail the nature and extent (%) of contribution by the PhD candidate:

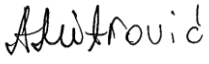
This was my own research. I conducted the evaluation studies presented in the paper. I also analysed the results. Prof. Mitrovic provided advice and reviewed the paper. My contribution is about 75%.

Certification by Co-authors:

If there is more than one co-author then a single co-author vcan sign on behalf of all

The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the PhD candidate was the lead author of the co-authored work he or she wrote the text

Name: *Tanja Mitrovic* Signature:  Date: *2.10.2012*